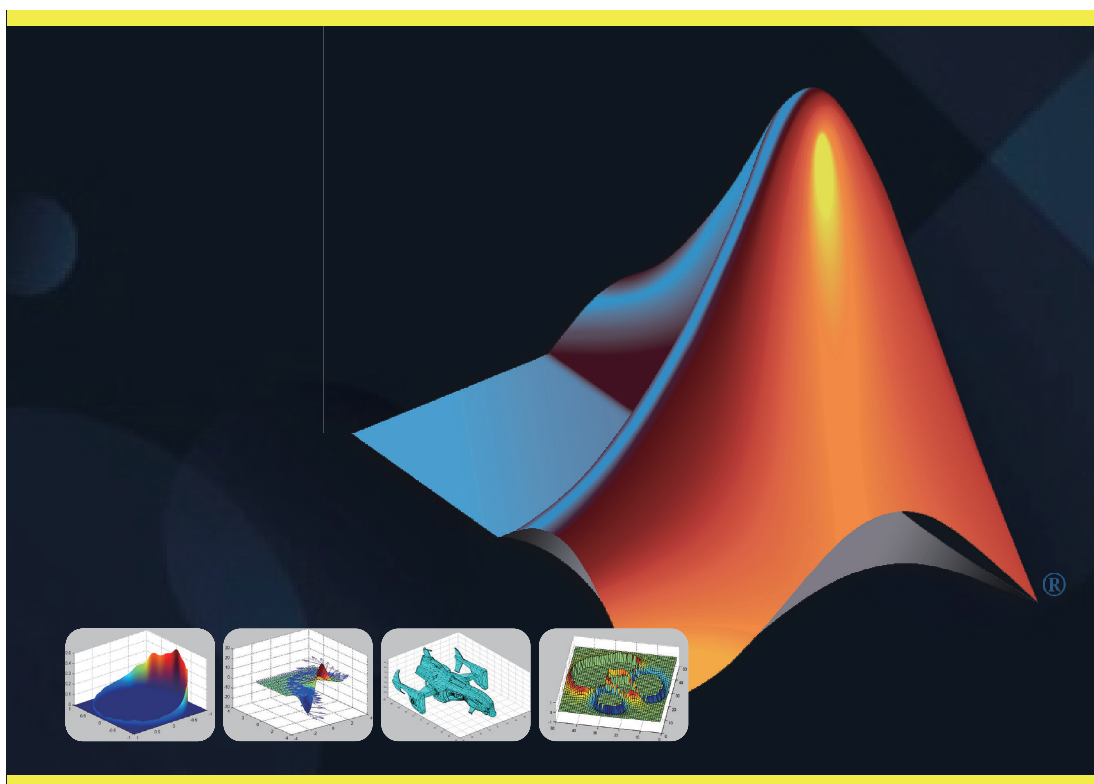
MATLAB R2018a 完全自学一本通

刘浩　韩晶　编著



電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

VISIT…

LANZAROTE
Caliente.com

内 容 简 介

本书面向 MATLAB 的初、中级读者，在介绍 MATLAB R2018a 集成环境的基础上，对 MATLAB 使用中常用的知识和工具进行了详细的介绍，书中各章均提供了大量有针对性的算例，供读者实战练习。

根据内容的侧重点不同，全书分为5部分共24章：第1～5章为基础部分；第6～10章为数学应用部分；第11～15章为工程应用部分；第16～20章为高级应用部分；第21～24章为知识拓展部分。

由于篇幅受限，第18～24章为电子书形式。本书附赠电子书及素材获取方法：扫描封底二维码，关注有艺公众号，在“有艺学堂”的“资源下载”中获取。

为了使用户能够更好地操作 MATLAB，本书中示例的命令已记录在 M 文件及其他相关文件中，用户可以将相关的目录设置为工作目录，直接使用 M 文件进行操作，以便快速掌握 MATLAB 的使用方法。

本书结构严谨、内容全面、图文并茂、实例丰富，既适合信号处理、通信工程、自动控制、机械电子、自动化、电力电气等专业的本科生、研究生、教师和科技工作者学习使用，也可以作为广大 MATLAB 爱好者的自学用书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据
MATLAB R2018a 完全自学一本通 / 刘浩，韩晶编著. -- 北京：电子工业出版社，2019.1
ISBN 978-7-121-35382-6

Ⅰ. ① M… Ⅱ. ①刘… ②韩… Ⅲ. ① Matlab 软件 Ⅳ. ① TP317

中国版本图书馆 CIP 数据核字 (2018) 第 252902 号

责任编辑：赵英华
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　　　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　印张：35　字数：899.2 千字
版　　次：2019 年 1 月第 1 版
印　　次：2019 年 1 月第 1 次印刷
定　　价：89.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
本书咨询联系方式：（010）88254161~88254167 转 1897。

前言
PREFACE

MATLAB R2018a 是 2018 年最新发行的 MATLAB 版本，它为数据分析与处理提供了强大的工具。目前，MATLAB 已经在很多领域取得了成功应用。在这些领域的成功应用表明，MATLAB 所代表的数据分析处理手段在科学、工程等方面将发挥重要的作用。

本书针对 MATLAB R2018a 在部分与数学高度相关领域内的应用，引导读者掌握 MATLAB 的应用。

本书特点

- 由浅入深，循序渐进：本书以初、中级读者为对象，首先从 MATLAB 的基础知识讲起，辅以 MATLAB 在工程中的应用案例，帮助读者尽快掌握 MATLAB 进行科学计算及工程分析的技能。
- 步骤详尽，内容新颖：本书结合作者多年 MATLAB 使用经验与实际工程应用案例，对 MATLAB 软件的使用方法与技巧进行详细讲解。本书内容新颖，讲解过程辅以相应的图片，使读者在阅读时一目了然，从而快速掌握书中所讲内容。
- 实例典型，轻松易学：通过学习实际工程应用案例的具体操作是掌握 MATLAB 最好的方式。本书通过应用案例，透彻、详尽地讲解 MATLAB 在各方面的应用。

本书内容

本书分为 5 部分共 24 章，面向 MATLAB 的初、中级读者。在介绍 MATLAB R2018a 集成环境的基础上，对 MATLAB 使用中常用的知识和工具进行详细的介绍，书中各章均提供大量有针对性的算例，供读者实战练习。

（1）第 1 ～ 5 章为基础部分，讲解 MATLAB R2018a 概述、MATLAB 基础知识、数组与矩阵、MATLAB 编程基础及数据可视化等。

第 1 章：MATLAB R2018a 概述　　第 2 章：MATLAB 基础知识
第 3 章：数组与矩阵　　第 4 章：MATLAB 编程基础
第 5 章：数据可视化

（2）第 6 ～ 10 章为数学应用部分，讲解数据分析、符号数学计算和概率统计等。

第 6 章：数据分析　　第 7 章：符号数学计算
第 8 章：概率统计　　第 9 章：数学建模函数
第 10 章：经典智能算法

（3）第 11 ～ 15 章为工程应用部分，讲解偏微分方程、优化、小波分析、Simulink 仿真基础及应用等。

第 11 章：偏微分方程工具箱　　第 12 章：优化工具箱

第 13 章：小波分析工具箱　　　　　　第 14 章：Simulink 仿真基础

第 15 章：Simulink 仿真应用

（4）第 16 ～ 20 章为高级应用部分，讲解句柄图形对象、图形用户界面、文件读取 I/O、MATLAB 编译器和外部接口应用介绍等内容。

第 16 章：句柄图形对象　　　　　　第 17 章：图形用户界面

第 18 章：文件读取 I/O　　　　　　第 19 章：MATLAB 编译器

第 20 章：外部接口应用介绍

（5）第 21 ～ 24 章为知识拓展部分，讲解图像处理工具箱、信号处理工具箱、神经网络工具箱、Stateflow 建模与应用等内容。

第 21 章：图像处理工具箱　　　　　　第 22 章：信号处理工具箱

第 23 章：神经网络工具箱　　　　　　第 24 章：Stateflow 建模与应用

由于篇幅受限，第 18 ～ 24 章为电子书形式。本书附赠电子书及素材获取方法：扫描封底二维码，关注有艺公众号，在“有艺学堂”的“资源下载”中获取。



读者对象

本书适合于 MALTAB 初学者和期望提高 MATLAB 数据分析及 Simulink 建模仿真工程应用能力的读者，具体说明如下：

◎ 初学 MATLAB 的技术人员　　　　◎ 广大科研工作人员

◎ 大中专院校的教师和在校生　　　◎ 相关培训机构的教师和学员

◎ 参加工作实习的“菜鸟”　　　　◎ MATLAB 爱好者



读者服务

为了方便解决本书疑难问题，读者在学习过程中遇到与本书有关的技术问题，可以发送邮件到邮箱 caxart@126.com，或者访问博客 http://blog.sina.com.cn/caxart，编者会尽快给予解答，我们将竭诚为您服务。



本书作者

参与本书编写的有丁金滨、刘浩、韩晶。虽然编者在本书的编写过程中力求叙述准确、完善，但由于水平有限，书中欠妥之处在所难免，希望读者和同仁能够及时指出，共同促进本书质量的提高。

最后再次希望本书能为读者的学习和工作提供帮助！

编　者

目录

CONTENTS

第 1 部分

第 1 章 MATLAB R2018a 概述............ 1

1.1 MATLAB R2018a 简介........................2

1.1.1 MathWorks 及其产品概述...............2

1.1.2 MATLAB 与其他数学软件............2

1.1.3 MATLAB 的主要特点....................3

1.1.4 MATLAB 的系统组成....................3

1.1.5 MATLAB 应用程序简介................4

1.1.6 MATLAB R2018a 新特性...............5

1.2 MATLAB R2018a 的目录结构...........6

1.3 MATLAB R2018a 的工作环境...........7

1.3.1 菜单 / 工具栏....................................7

1.3.2 命令行窗口.......................................8

1.3.3 工作区...9

1.4 MATLAB R2018a 的通用命令...........9

1.5 MATLAB R2018a 的文件管理.........11

1.5.1 当前文件夹浏览器和路径管理器..11

1.5.2 搜索路径及其设置........................11

1.6 MATLAB R2018a 的帮助系统.........13

1.6.1 纯文本帮助.....................................13

1.6.2 演示（Demos）帮助....................14

1.6.3 帮助导航浏览器............................14

1.7 初步使用 MATLAB..........................14

1.8 本章小结..15

第 2 章 MATLAB 基础知识................17

2.1 数据类型..18

2.1.1 数值类型...18

2.1.2 逻辑类型...22

2.1.3 字符和字符串................................24

2.1.4 函数句柄...26

2.1.5 结构体类型.....................................28

2.1.6 数组类型...30

2.1.7 单元数组类型................................32

2.1.8 map 容器类型.................................34

2.2 基本矩阵操作....................................38

2.2.1 矩阵和数组的概念及其区别.........38

2.2.2 矩阵的构造.....................................39

2.2.3 矩阵大小及结构的改变................43

2.2.4 矩阵下标引用................................44

2.2.5 矩阵信息的获取............................46

2.2.6 矩阵的保存和加载........................51

2.3 运算符..56

2.3.1 算术运算符.....................................56

2.3.2 关系运算符.....................................58

2.3.3 逻辑运算符.....................................59

2.3.4 运算优先级.....................................60

2.4 字符串处理函数................................60

2.4.1 字符串的构造................................61

2.4.2 字符串比较函数............................62

2.4.3 字符串查找和替换函数................63

2.4.4 字符串——数值转换....................64

2.5 本章小结..64

第 3 章 数组与矩阵.............................65

3.1 数组运算..66

3.1.1 数组的创建和操作........................66

3.1.2 数组的常见运算............................69
3.2 矩阵操作..72
3.2.1 创建矩阵..72
3.2.2 改变矩阵大小................................76
3.2.3 重构矩阵..78
3.3 矩阵元素的运算................................79
3.3.1 矩阵的加减运算............................79
3.3.2 矩阵的乘法运算............................80
3.3.3 矩阵的除法运算............................81
3.3.4 矩阵的幂运算................................82
3.3.5 矩阵元素的查找............................83
3.3.6 矩阵元素的排序............................83
3.3.7 矩阵元素的求和............................84
3.3.8 矩阵元素的求积............................85
3.3.9 矩阵元素的差分............................86
3.4 矩阵运算..86
3.4.1 矩阵分析..87
3.4.2 矩阵分解..91
3.4.3 特征值和特征向量........................97
3.5 稀疏矩阵..99
3.5.1 稀疏矩阵的存储方式....................99
3.5.2 稀疏矩阵的生成..........................100
3.5.3 稀疏矩阵的运算..........................104
3.6 本章小结..104

第 4 章 MATLAB 编程基础............... 105

4.1 M 文件编辑器..................................106
4.2 变量..108
4.2.1 变量的命名..................................108
4.2.2 变量的类型..................................109
4.2.3 MATLAB 默认的特殊变量.........109
4.2.4 关键字..110
4.3 MATLAB 的控制流........................110
4.3.1 顺序结构......................................110
4.3.2 if-else-end 分支结构...................111
4.3.3 switch-case 结构.........................112
4.3.4 try-catch 结构.............................115
4.3.5 for 循环结构................................115
4.3.6 while 循环结构............................117
4.4 控制程序流的其他常用指令...........118
4.4.1 return 指令..................................118
4.4.2 input 和 keyboard 指令...............118
4.4.3 yesinput 指令..............................119
4.4.4 pause 指令...................................119
4.4.5 continue 指令..............................119
4.4.6 break 指令...................................120
4.4.7 error 和 warning 指令..................120
4.5 脚本和函数......................................120
4.5.1 脚本..120
4.5.2 函数..121
4.5.3 M 文件的一般结构......................123
4.5.4 匿名函数、子函数、私有函数与私有目录..................................125
4.5.5 重载函数......................................126
4.5.6 eval 和 feval 函数........................126
4.5.7 内联函数......................................129
4.5.8 向量化和预分配..........................131
4.5.9 函数的函数..................................132
4.5.10 P 码文件.....................................133
4.6 M 文件中变量的检测与传递..........134
4.6.1 输入 / 输出变量检测指令...........134
4.6.2 “可变数量”输入 / 输出变量...135
4.6.3 跨空间变量传递和赋值...............137
4.7 MATLAB 程序的调试....................139
4.7.1 程序调试的基本概念...................139
4.7.2 直接调试法...................................140
4.7.3 使用调试函数进行调试...............140
4.7.4 工具调试法...................................143
4.7.5 程序的性能优化技术...................144
4.8 本章小结..146

第 5 章 数据可视化........................... 147

5.1 图形绘制..148

5.1.1 离散数据及离散函数..................148

5.1.2 连续函数......................................148

5.1.3 图形绘制示例..............................149

5.1.4 图形绘制的基本步骤..................151

5.2 二维图形绘制..................................151

5.2.1 plot 指令 151

5.2.2 格栅..155

5.2.3 文字说明......................................156

5.2.4 线型、标记和颜色......................159

5.2.5 坐标轴设置..................................161

5.2.6 图形叠绘......................................162

5.2.7 子图绘制......................................163

5.2.8 交互式绘图..................................164

5.2.9 双坐标轴绘制..............................165

5.2.10 fplot 绘图指令...........................166

5.2.11 ezplot 绘图指令.........................167

5.2.12 特殊坐标轴绘图........................168

5.2.13 二维特殊图形函数....................170

5.3 三维图形绘制..................................174

5.3.1 曲线图绘制..................................174

5.3.2 网格图绘制..................................175

5.3.3 曲面图绘制..................................176

5.3.4 光照模型......................................177

5.3.5 绘制等值线图..............................178

5.4 四维图形可视化..............................179

5.4.1 用颜色描述第四维......................179

5.4.2 其他函数......................................180

5.5 本章小结..181

第 2 部分

第 6 章 数据分析............................... 183

6.1 多项式及其函数..............................184

6.1.1 多项式的表达式和创建..............184

6.1.2 多项式求根..................................185

6.1.3 多项式的四则运算......................186

6.1.4 多项式的导数、积分与估值.......188

6.1.5 多项式运算函数及操作指令.......189

6.1.6 有理多项式..................................189

6.2 数据插值..191

6.2.1 一维插值......................................191

6.2.2 二维插值......................................195

6.3 函数的极限......................................196

6.3.1 极限的概念..................................196

6.3.2 求极限的函数..............................196

6.4 函数数值积分..................................197

6.4.1 数值积分问题的数学表述..........198

6.4.2 一元函数的数值积分..................198

6.4.3 多重数值积分..............................200

6.5 本章小结..201

第 7 章 符号数学计算........................ 203

7.1 MATLAB 符号计算概述 204

7.2 符号对象和符号表达式..................204

7.2.1 符号对象的创建命令..................204

7.2.2 符号对象的创建示例..................205

7.2.3 符号计算中的运算符和函数.......208

7.2.4 符号对象的类别识别函数..........211

7.2.5 符号表达式中的变量确定..........213

7.2.6 符号精度计算..............................214

7.3 符号表达式操作..............................216

7.3.1 符号表达式显示..........................216

7.3.2 符号表达式合并..........................216

7.3.3 符号表达式展开..........................217

7.3.4 符号表达式嵌套..........................218

7.3.5 符号表达式分解..........................218

7.3.6 符号表达式简化..........................219

7.4 符号表达式替换..............................221

7.4.1 subs 替换函数 221
7.4.2 subexpr 替换函数 222
7.5 符号函数的操作 223
7.5.1 复合函数操作 223
7.5.2 反函数操作 224
7.6 符号微积分 225
7.6.1 符号表达式的极限 225
7.6.2 符号表达式的微分 226
7.6.3 符号表达式的积分 227
7.6.4 符号表达式的级数求和 228
7.6.5 符号表达式的泰勒级数 229
7.7 符号积分变换 230
7.7.1 傅里叶变换及其反变换 230
7.7.2 拉普拉斯变换及其反变换 231
7.7.3 Z 变换及其反变换 232
7.8 符号代数方程求解 233
7.9 符号微分方程求解 235
7.10 符号分析可视化 237
7.10.1 funtool 分析界面 237
7.10.2 taylortool 分析界面 240
7.11 本章小结 241

第 8 章 概率统计 243

8.1 产生随机变量 244
8.1.1 二项分布的随机数据的产生 244
8.1.2 正态分布的随机数据的产生 244
8.1.3 常见分布的随机数产生 245
8.2 概率密度计算 246
8.2.1 通用函数概率密度值 246
8.2.2 专用函数概率密度值 248
8.3 累积概率分布 249
8.3.1 通用函数累积概率值 249
8.3.2 专用函数累积概率值 250
8.4 统计特征 251
8.4.1 平均值、中值 251
8.4.2 数据比较 252
8.4.3 期望 253
8.4.4 方差和标准差 253
8.4.5 协方差与相关系数 253
8.5 统计作图 255
8.5.1 正整数频率表 255
8.5.2 累积分布函数图形 255
8.5.3 最小二乘拟合直线 256
8.5.4 绘制正态分布概率图形 257
8.5.5 样本数据的盒图 258
8.5.6 参考线绘制 258
8.5.7 样本概率图形 260
8.5.8 正态拟合直方图 261
8.6 本章小结 262

第 9 章 数学建模函数 263

9.1 曲线拟合函数 264
9.1.1 多项式拟合 264
9.1.2 加权最小方差（WLS）拟合
原理及实例 265
9.1.3 非线性曲线拟合 267
9.2 参数估计函数 268
9.2.1 常见分布的参数估计 268
9.2.2 点估计 270
9.2.3 区间估计 272
9.3 参数传递 272
9.3.1 输入和输出参数的数目 273
9.3.2 可变数目的参数传递 274
9.3.3 返回被修改的输入参数 274
9.3.4 全局变量 276
9.4 本章小结 276

第 10 章 经典智能算法 277

10.1 粒子群算法的 MATLAB 实现 278
10.1.1 基本原理 278
10.1.2 程序设计 279

10.1.3 经典应用....................................286
10.2 遗传算法的 MATLAB 实现 290
10.2.1 基本原理....................................290
10.2.2 程序设计....................................293
10.2.3 经典应用....................................297
10.3 蚁群算法概述................................303
10.3.1 基本原理....................................303
10.3.2 程序设计....................................304
10.3.3 经典应用....................................306
10.4 本章小结..314

第 3 部分

第 11 章 偏微分方程工具箱................ 315
11.1 偏微分方程（PDE）工具箱介绍..316
11.1.1 偏微分方程常见类型介绍.........316
11.1.2 偏微分方程工具箱简介............322
11.2 求解偏微分方程示例....................323
11.2.1 求解椭圆方程.............................324
11.2.2 求解抛物线方程.........................328
11.2.3 求解双曲线方程.........................331
11.2.4 求解特征值方程.........................335
11.3 本章小结..338
第 12 章 优化工具箱 339
12.1 优化工具箱及最优化问题介绍.....340
12.1.1 优化工具箱常用函数................340
12.1.2 最优化问题................................343
12.2 线性规划..345
12.2.1 线性规划函数............................346
12.2.2 线性规划问题的应用................347
12.3 无约束非线性规划........................355
12.3.1 基本数学原理介绍....................355
12.3.2 无约束非线性规划函数............356
12.3.3 无约束非线性规划问题的应用..359
12.4 二次规划..359
12.4.1 二次规划函数 quadprog 359
12.4.2 二次规划问题的应用................361
12.5 有约束最小化................................362
12.5.1 有约束最小化函数 fmincon 362
12.5.2 有约束最小化应用....................365
12.6 目标规划..366
12.6.1 目标规划函数 fgoalattain 366
12.6.2 目标规划应用............................369
12.7 最大最小化....................................372
12.7.1 最大最小化函数 fminimax 372
12.7.2 最大最小化应用........................374
12.8 本章小结..375
第 13 章 小波分析工具箱................... 377
13.1 小波分析..378
13.1.1 小波分析概述............................378
13.1.2 傅里叶变换和小波变换............379
13.1.3 多分辨分析................................382
13.1.4 小波包分析................................385
13.1.5 几种常用的小波........................389
13.1.6 小波应用演示示例....................391
13.2 小波工具箱介绍............................392
13.2.1 小波工具箱的启动....................392
13.2.2 一维连续小波分析工具............393
13.2.3 一维离散小波分析工具............398
13.3 小波分析在图像处理中的应用.....403
13.3.1 基于小波变换的图像局部压缩..404
13.3.2 二维小波变换的图像压缩.........405
13.3.3 图像压缩中阈值的确定与作用命令...................................408
13.3.4 基于小波包变换的图像压缩.....409

13.3.5 小波变换用于图像去噪............411
13.3.6 小波分析用于图像增强............414
13.3.7 图像钝化与锐化........................416
13.3.8 小波分析用于图像融合............418
13.3.9 小波分析用于图像分解............420
13.4 本章小结.......................................422

第 14 章 Simulink 仿真基础............... 423

14.1 Simulink 概述...............................424
14.1.1 Simulink 的基本概念................424
14.1.2 Simulink 的工作环境与启动.....426
14.1.3 Simulink 模型的特点................427
14.1.4 Simulink 模块的组成................429
14.1.5 Simulink 中的数据类型............430
14.1.6 Simulink 中的模块和模块库.....433
14.1.7 Simulink 常用工具....................438
14.1.8 Simulink 的实例演示................439
14.2 Simulink 模型创建........................441
14.2.1 Simulink 模块的基本操作.........442
14.2.2 模型和模型文件........................445
14.2.3 创建模型的基本步骤和技巧.....447
14.3 过零检测和代数环........................450
14.3.1 过零检测....................................450
14.3.2 代数环..452
14.4 本章小结.......................................453

第 15 章 Simulink 仿真应用............... 455

15.1 子系统的创建和封装....................456
15.1.1 子系统介绍................................456
15.1.2 创建子系统................................457
15.1.3 封装子系统................................458
15.2 仿真模型的分析...........................462
15.2.1 模型状态的确定........................462
15.2.2 线性化的数学描述....................464
15.2.3 平衡点分析................................465
15.2.4 微分方程的求解算法................466
15.3 运行仿真.......................................467
15.3.1 启动仿真....................................467
15.3.2 仿真的配置................................470
15.3.3 优化仿真过程............................475
15.4 模型调试.......................................477
15.4.1 Simulink 调试器........................477
15.4.2 命令行调试及设置断点............478
15.5 Simulink 实例应用........................479
15.6 本章小结.......................................484

第 4 部分

第 16 章 句柄图形对象..................... 485

16.1 句柄图形对象体系........................486
16.1.1 句柄图形组织............................486
16.1.2 句柄图形对象类型简介............486
16.2 句柄图形对象操作........................487
16.2.1 创建对象....................................487
16.2.2 访问对象句柄............................489
16.2.3 复制和删除对象........................491
16.2.4 控制图形输出............................492
16.2.5 保存句柄....................................496
16.3 对象属性设置...............................496
16.3.1 设置属性....................................497
16.3.2 设置默认属性............................498
16.3.3 通用属性....................................500
16.4 Figure 对象...................................501
16.4.1 Figure 对象介绍........................501
16.4.2 Figure 对象操作........................502
16.5 Axes 对象......................................502
16.5.1 标签与外观................................503
16.5.2 位置..503
16.5.3 一图多轴....................................504
16.5.4 坐标轴控制................................507

16.5.5 线条颜色控制............................509
16.5.6 绘图操作....................................509
16.6 Core 对象..510
16.6.1 Core 对象介绍............................510
16.6.2 Core 对象创建示例....................511
16.7 Plot 对象..512
16.7.1 Plot 对象介绍............................512
16.7.2 Plot 对象创建示例....................512
16.7.3 连接变量....................................513
16.8 Group 对象......................................513
16.8.1 Group 对象介绍.........................513
16.8.2 创建 Group 对象........................514
16.8.3 对象变换....................................514
16.9 Annotation 对象..............................515
16.9.1 Annotation 对象介绍.................515
16.9.2 Annotation 对象使用示例..........515
16.10 本章小结......................................516

第 17 章 图形用户界面......................517

17.1 图形用户界面介绍.........................518
17.2 图形用户界面控件.........................518
17.2.1 图形用户界面控件的创建.........519
17.2.2 鼠标动作执行............................521
17.2.3 事件队列的执行顺序.................522
17.2.4 回调函数的编写.........................523
17.3 对话框对象....................................525
17.3.1 公共对话框................................526
17.3.2 一般对话框................................529
17.4 界面菜单..535
17.4.1 菜单建立....................................535
17.4.2 菜单属性....................................536
17.5 编写 M 文件..................................537
17.6 图形界面创建工具 GUIDE...........539
17.6.1 利用创建工具 GUIDE 进行 GUI 设计...................................539
17.6.2 GUIDE 创建的工具..................545
17.6.3 创建带 Uicontrol 控件的图形界面....................................547
17.7 本章小结..548

第 1 部分

第 1 章

MATLAB R2018a 概述

知识要点

MATLAB R2018a 是 MathWorks 公司发布的最新版的集算法开发、数据可视化、数据分析及数值计算于一体的高级技术计算语言和交互式环境。本章将介绍 MATLAB R2018a 的最新特性、MATLAB 界面功能及如何对界面的各部分进行操作。

学习要求	知识点 \ 学习目标	了解	理解	应用	实践
	MATLAB 产品及发展历史	√			
	MATLAB R2018a 的新特性	√			
	MATLAB R2018a 的目录结构	√			
	MATLAB R2018a 的工作环境			√	√
	MATLAB R2018a 的文件管理		√	√	√
	MATLAB R2018a 的帮助系统		√		√

1.1 MATLAB R2018a 简介

MathWorks 公司创立于 1984 年，总部位于美国马萨诸塞州内迪克，在全球 15 个国家有 2 200 名员工，是领先的为工程师和科学家提供数学计算软件的供应商，旗下的产品包括 MATLAB 产品家族、Simulink 产品家族及 PolySpace 产品家族。

1.1.1 MathWorks 及其产品概述

MATLAB 是矩阵实验室（Matrix Laboratory）的简称，是一种用于算法开发、数据可视化、数据分析及数值计算的高级技术计算语言和交互式环境。MATLAB 的应用范围非常广，包括信号和图像处理、通信、控制系统设计、测试和测量、财务建模和分析，以及计算生物学等众多应用领域。附加的工具箱（单独提供的专用 MATLAB 函数集）扩展了 MATLAB 的使用环境，以解决这些应用领域内特定类型的问题。

Simulink 是一个用于对动态系统进行多域建模和模型设计的平台。它提供了一个交互式图形环境，以及一个自定义模块库，并可针对特定应用加以扩展，可应用于控制系统设计、信号处理和通信及图像处理等众多领域。

PolySpace 提供代码验证，可确保消除源代码中的溢出、除零、数组访问越界及其他运行错误。此类产品可以证明源代码中不存在某些运行错误，使工程师能够选择并跟踪嵌入式软件质量的指标和阈值，帮助软件团队更好地定义质量目标，并更快地实施。该软件已经在汽车、航空、国防及工业自动化和机械行业中得到广泛应用。

目前，经常使用的 MATLAB 软件集成了 MATLAB 和 Simulink 两大模块，也是本书即将介绍的内容；而 PolySpace 是独立于 MATLAB 软件之外的，需要单独安装。

1.1.2 MATLAB 与其他数学软件

除 MATLAB 外，其他广泛应用的数学软件很多，著名的有 Mathematica 和 Maple。

Mathematica 是一个综合的数学软件环境，具有数值计算、符号推导、数据可视化和编程等多种功能，在符号计算领域有很高的知名度。整个 Mathematica 软件分为两大部分：Kemel 和 FrontEnd。Kemel 是软件的计算中心，而 FrontEnd 负责与用户的交流，两者有一定的独立性。Mathematica 的表达式含义十分丰富，几乎包含了一切要处理的对象。

Maple 是当今世界上较优秀的几款数学软件之一。它以友善的使用环境、强大的符号处理、精确的数值计算、灵活的图形显示、高效的编程功能为越来越多的教师、学生和科研人员所喜爱，并成为他们进行数学处理的首选工具。由于 Maple 软件原是为符号计算而设计的，因此在数值计算与绘图方面的运算速度要比 MATLAB 慢。Maple 的帮助系统是用英语写的，这给英文差的人们带来了不便。

MATLAB 作为和 Mathematica、Maple 并列的三大数学软件之一，强项就是其强大的矩

阵计算及仿真能力。每次 MathWorks 公司发布 MATLAB 的同时，也会发布仿真工具 Simulink。

在欧美国家，很多大公司将产品投入实际使用之前都会进行仿真试验，他们主要使用的仿真软件就是 Simulink。MATLAB 提供了自己的编译器，全面兼容 C++及 Fortran 两大语言。因此，MATLAB 成为工程师、科研工作者最好的语言、最好的工具。

1.1.3　MATLAB 的主要特点

MATLAB 以其良好的开放性和运行的可靠性，已经成为国际控制界公认的标准计算软件，在国际上 30 多个数学类科技应用软件中，MATLAB 在数值计算方面独占鳌头。

（1）计算功能强大。

（2）绘图非常方便。在 Fortran 和 C 语言里，绘图都很不容易，但在 MATLAB 里，数据的可视化非常简单。而且，MATLAB 还具有较强的编辑图形界面的能力。

（3）功能强大的工具箱是 MATLAB 的另一特色。MATLAB 包含两个部分：核心部分和各种可选的工具箱。核心部分有数百个核心内部函数。其工具箱又分为两类：功能性工具箱和学科性工具箱。

功能性工具箱主要用来扩充其符号计算功能、图示建模仿真功能、文字处理功能及与硬件实时交互功能。功能性工具箱用于多种学科。

学科性工具箱的专业性比较强，如 control、signal processing、communication 等。这些工具箱都是由该领域内学术水平很高的专家编写的，所以用户无须编写自己学科范围内的基础程序，而直接进行高、精、尖的研究即可。

除内部函数以外，MATLAB 的所有核心文件和工具箱文件都是可读可写的源文件，用户可通过对源文件的修改及加入自己的文件构成新的工具箱。

（4）帮助功能完整：自带的帮助功能是非常强大的帮助手册。

1.1.4　MATLAB 的系统组成

MATLAB 系统由 MATLAB 开发环境、MATLAB 数学函数库、MATLAB 语言、MATLAB 图形处理系统和 MATLAB 应用程序接口（API）五大部分构成。

1．MATLAB 开发环境

MATLAB 开发环境是一套方便用户使用 MATLAB 函数和文件的工具集，其中许多工具是图形化用户接口。它是一个集成化的工作区，可以让用户输入、输出数据，并提供了 M 文件的集成编译和调试环境。它包括 MATLAB 桌面、命令行窗口、M 文件编辑调试器、MATLAB 工作区和在线帮助文档等。

2．MATLAB 数学函数库

MATLAB 数学函数库包括了大量的计算算法，从基本运算（如加法）到复杂算法（如矩阵求逆、贝济埃函数、快速傅里叶变换等），体现了其强大的数学计算功能。

3．MATLAB 语言

MATLAB 语言是一个高级的基于矩阵/数组的语言，包括程序流控制、函数、脚本、数据结构、输入/输出、工具箱和面向对象编程等特色。用户既可以用它来快速编写简单的程序，也可以用它来编写庞大复杂的应用程序。

4．MATLAB 图形处理系统

图形处理系统使得 MATLAB 能方便地图形化显示向量和矩阵，而且能对图形添加标注和打印。它包括强力的二维及三维图形函数、图像处理和动画显示等函数。

5．MATLAB 程序接口

MATLAB 程序接口可以使 MATLAB 方便地调用 C 和 Fortran 程序，以及在 MATLAB 与其他应用程序间建立客户/服务器关系。

1.1.5 MATLAB 应用程序简介

应用程序（也称工具箱）是 MATLAB 的重要部分，它是 MATLAB 强大功能得以实现的载体和手段，是对 MATLAB 基本功能的重要扩充。

> **提示**
>
> MATLAB 会不定时更新应用程序，读者可到 http://www.mathworks.com/products 中了解 MATLAB 应用程序的最新动态。

应用程序又可以分为功能性应用程序和学科应用程序。功能性应用程序用来扩充 MATLAB 的符号计算、可视化建模仿真，以及与硬件实时交互等功能，能用于多种学科；学科应用程序是专业性比较强的应用程序，控制工具箱、信号处理与通信工具箱等都属于此类。

在 MATLAB R2018a 版本中应用程序如图 1-1 所示。下面将科学计算中常用的应用程序所包含的主要内容进行简单介绍。

1．样条工具箱

- 分段多项式和 B 样条。
- 样条的构造。
- 曲线拟合及平滑。
- 函数微积分。

2．优化工具箱

- 线性规划和二次规划。
- 求函数的最大值和最小值。
- 多目标优化。
- 约束条件下的优化。
- 非线性方程求解。

3．偏微分方程工具箱

- 二维偏微分方程的图形处理。

- 几何表示。
- 自适应曲面绘制。
- 有限元方法。

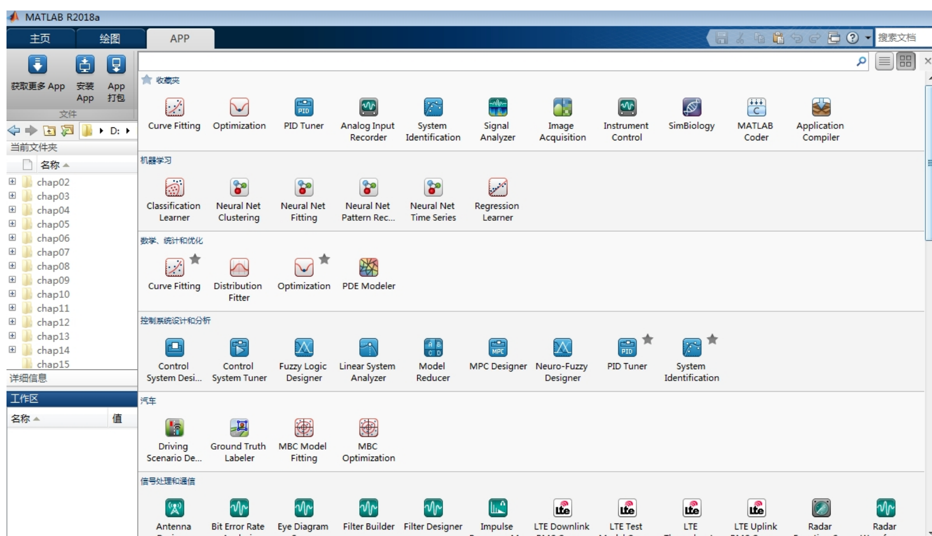


图 1-1　展开的工具箱

1.1.6　MATLAB R2018a 新特性

MATLAB R2018a 包括 MATLAB 和 Simulink 产品的新功能，以及其他产品的更新和补丁修复。

（1）MATLAB 产品系列的新功能如下：

- Optimization Toolbox：混合整数线性规划（MILP）求解器。
- Statistics Toolbox：使用每个主题的多个测量值反复测量数据建模。
- Image Processing Toolbox：使用 MATLAB Coder 为 25 个函数生成 C 代码，为 5 个函数增加了 GPU 加速功能。
- Econometrics Toolbox：状态空间模型、缺失数据情况下自校准的卡尔曼滤波器，以及 ARIMA/GARCH 模型性能增强。
- Financial Instruments Toolbox：对偶曲线构建，用于计算信用敞口和敞口概况的函数，以及利率上限、利率下限和掉期期权的布莱克模型定价。
- SimBiology：提供用于模型开发的模型估算和桌面增强的统一函数。
- System Identification Toolbox：递归最小二乘估算器和在线模型参数估算模块。
- MATLAB Production Server：实现客户端与服务器之间的安全通信，以及动态请求创建。

（2）Simulink 产品系列的新功能如下：

- Simulink：用于定义和管理与模型关联的设计数据的数据字典。
- Simulink：用于算法分割和用于多核处理器及 FPGA 目标器件的单一模型工作流程。

- Simulink：为 LEGO MINDSTORMS EV3、Arduino Due 和 Samsung Galaxy Android 设备提供内建支持。
- Stateflow：在 Stateflow 状态图中填写上下文相关的选项卡。
- Simulink Real-Time：仪表板、高分辨率目标显示器和 FlexRay 协议支持，以及合并了 xPC Target 和 xPC Target Embedded Option 的功能。
- SimMechanics：STEP 文件导入和接头的总约束力计算。
- Simulink Report Generator：用于在 Simulink Web View 视图中丰富显示内容的对象检查器和通知程序。

除此之外，MATLAB R2018a 版本有以下几个亮点：

- MATLAB 实时编辑器：提供一种全新方式来创建、编辑和运行 MATLAB 代码，加快探索性编程和分析的速度。
- App Designer：提供增强的设计环境和 UI 组件集，用于构建 MATLAB 应用程序的环境，简化了布置用户界面可视组件的过程。
- 深度学习：用于图像分类问题。
- Simulink Start Page：通过访问模板、最新模型以及精选示例更快地开始或继续工作。
- SimEvents 新引擎：创建包含事件操作和新模块的离散事件模型和调度程序。
- 飞行仪器库：使用标准座舱仪器显示飞行条件。

1.2 MATLAB R2018a 的目录结构

当用户在计算机上成功安装了 MATLAB R2018a 后，在用户自定义的安装目录内便包含一系列的文件和文件夹，如图 1-2 所示。

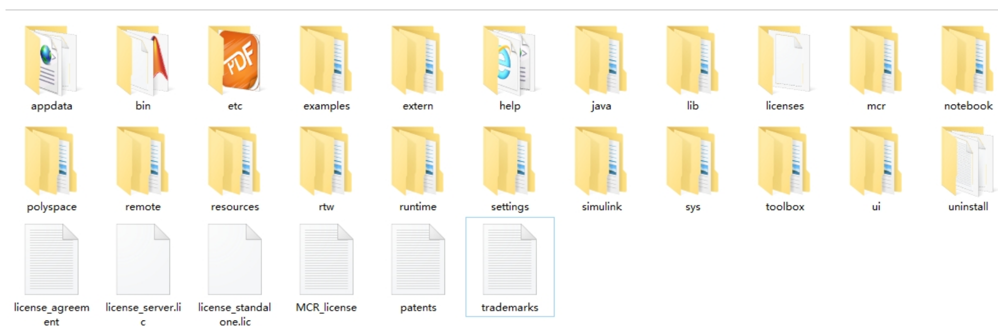


图 1-2 MATLAB R2018a 安装目录下的文件和文件夹

下面重点介绍其中部分文件和文件夹的用途。

- \bin\win32：MATLAB R2018a 系统中可执行的相关文件。
- \extern：创建 MATLAB R2018a 的外部程序接口的工具。
- \help：帮助系统。
- \java：MATLAB R2018a 的 Java 支持程序。

- \notebook：Notebook 是用来实现 MATLAB 教学工作环境与 Word 文字处理环境信息交互的软件，是一个兼备数学计算、图形显示和文字处理能力的集成环境。
- \rtw：Real-Time Workshop 软件包。
- \simulink：Simulink 软件包，用于动态系统的建模、仿真与分析。
- \stateflow：Stateflow 软件包，用于状态机设计的图形化开发和设计工具。
- \sys：MATLAB R2018a 需要的工具和操作系统库。
- \toolbox：MATLAB R2018a 的各种应用程序。
- \uninstall：MATLAB R2018a 的卸载程序。
- \license.txt：软件许可协议的内容。
- \patents.txt：软件申请的专利内容。

1.3　MATLAB R2018a 的工作环境

在 MATLAB R2018a 的安装目录内的 bin 文件夹下，双击 MATLAB.exe 图标，启动 MATLAB R2018a，出现启动界面，如图 1-3 所示；启动后，弹出 MATLAB R2018a 的用户界面。

MATLAB R2018a 的主界面即用户的工作环境，包括菜单栏、工具栏、开始按钮和各个不同用途的窗口，如图 1-4 所示。本节主要介绍 MATLAB 各交互界面的功能及其操作。

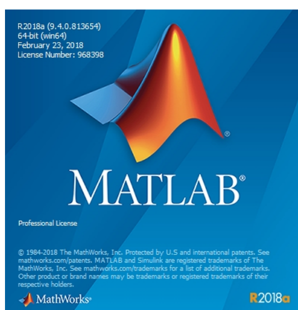


图 1-3　MATLAB R2018a 启动界面

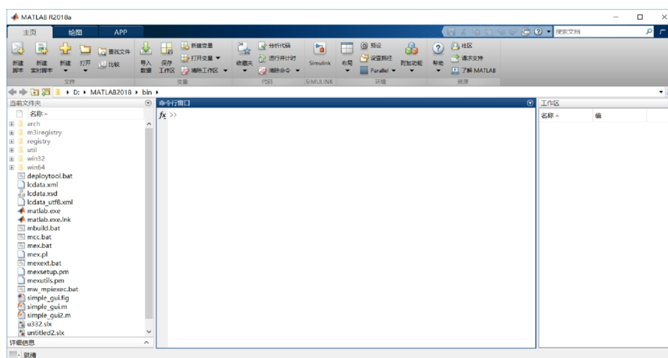


图 1-4　MATLAB R2018a 的主界面

1.3.1　菜单/工具栏

MATLAB 的菜单/工具栏中包含 3 个标签，分别为主页、绘图和应用程序。其中，绘图标签下提供数据的绘图功能；而应用程序标签则提供了各应用程序的入口。主页标签提供了下述主要功能。

- 新建：用于建立新的.m 文件、图形、模型和图形用户界面。
- 新建脚本：用于建立新的.m 脚本文件。
- 打开：用于打开 MATLAB 的.m 文件、.fig 文件、.mat 文件、.mdl 文件、.cdr 文件等，也可通过快捷键 Ctrl+O 来实现此项操作。

- 导入数据：用于从其他文件导入数据，单击后弹出对话框，选择导入文件的路径和位置。
- 保存工作区：用于把工作区的数据存放到相应的路径文件中。
- 设置路径：设置工作路径。
- 预设：用于设置命令窗的属性，单击 预设 按钮弹出如图 1-5 所示的对话框。
- 布局：提供工作界面上各个组件的显示选项，并提供预设的布局。
- 帮助：打开帮助文件或其他帮助方式。

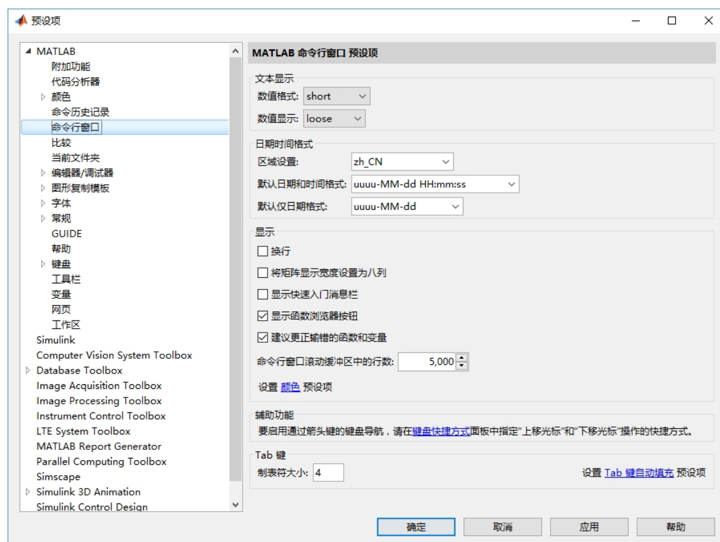


图 1-5 “预设项”对话框

1.3.2 命令行窗口

命令行窗口是 MATLAB 最重要的窗口。用户输入各种指令、函数、表达式等，都是在命令行窗口内完成的，如图 1-6 所示。

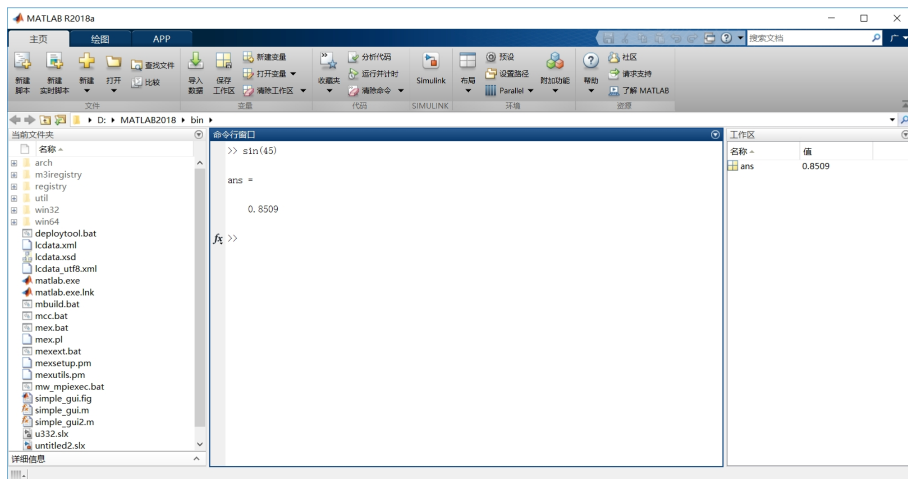


图 1-6 命令行窗口

> 注意
>
> “>>”是运算提示符，表示 MATLAB 处于准备状态，等待用户输入指令进行计算。当在提示符后输入命令，并按 Enter 键确认后，MATLAB 会给出计算结果，并再次进入准备状态。

单击命令行窗口右上角的下三角形图标并选择“取消停靠”，可以使命令行窗口脱离 MATLAB 界面成为一个独立的窗口；同理，单击独立的命令行窗口右上角的下三角形图标并选择“停靠”，可使命令行窗口再次合并到 MATLAB 主界面。

1.3.3　工作区

工作区窗口显示当前内存中所有的 MATLAB 变量的变量名、数据结构、字节数及数据类型等信息，如图 1-7 所示。不同的变量类型分别对应不同的变量名图标。

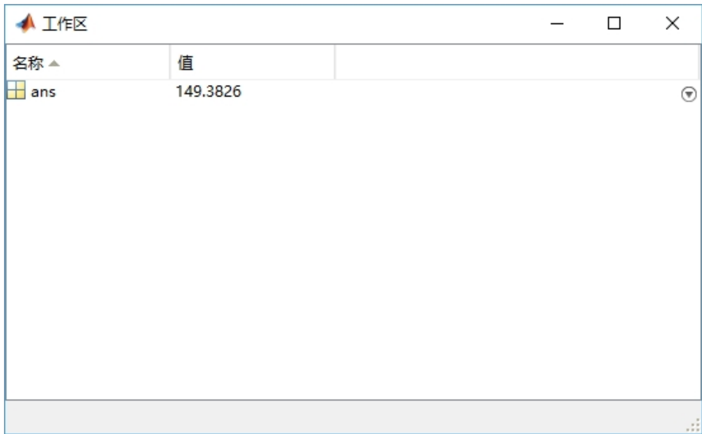


图 1-7　工作区窗口

用户可以选中已有变量，单击鼠标右键对其进行各种操作。此外，工作界面的菜单/工具栏上也有相应的命令供用户使用。

- 新建变量：向工作区添加新的变量。
- 导入数据：向工作区导入数据文件。
- 保存工作区：保存工作区中的变量。
- 清除工作区：删除工作区中的变量。

1.4　MATLAB R2018a 的通用命令

通用命令是 MATLAB 中经常使用的一组命令，这些命令可以用来管理目录、命令、函数、变量、工作区、文件和窗口。为了更好地使用 MATLAB，用户需要熟练掌握和理解这些命令。下面对这些命令进行介绍。

1．常用命令

常用命令的功能如表 1-1 所示。

表 1-1　常用命令

命　　令	命令说明	命　　令	命令说明
cd	显示或改变当前工作文件夹	load	加载指定文件的变量
dir	显示当前文件夹或指定目录下的文件	diary	日志文件命令
clc	清除工作窗中的所有显示内容	!	调用 DOS 命令
home	将光标移至命令行窗口的最左上角	exit	退出 MATLAB
clf	清除图形窗口	quit	退出 MATLAB
type	显示文件内容	pack	收集内存碎片
clear	清理内存变量	hold	图形保持开关
echo	工作窗信息显示开关	path	显示搜索目录
disp	显示变量或文字内容	save	保存内存变量到指定文件

2．输入内容的编辑

在命令行窗口中，为了便于对输入的内容进行编辑，MATLAB R2018a 提供了一些控制光标位置和进行简单编辑的常用编辑键与组合键，掌握这些可以在输入命令的过程中起到事半功倍的效果。表 1-2 列出了一些常用键盘按键及其作用。

表 1-2　命令行中的键盘按键

键盘按键	说　　明	键盘按键	说　　明
↑	Ctrl+P，调用上一行	Home	Ctrl+A，光标置于当前行开头
↓	Ctrl+N，调用下一行	End	Ctrl+E，光标置于当前行末尾
←	Ctrl+B，光标左移一个字符	Esc	Ctrl+U，清除当前输入行
→	Ctrl+F，光标右移一个字符	Delete	Ctrl+D，删除光标处的字符
Ctrl+←	Ctrl+L，光标左移一个单词	Backspace	Ctrl+H，删除光标前的字符
Ctrl+→	Ctrl+R，光标右移一个单词	Alt+Backspace	恢复上一次删除

3．标点

在 MATLAB 语言中，一些标点符号也被赋予了特殊的意义或代表一定的运算，具体内容如表 1-3 所示。

表 1-3　MATLAB 语言的标点

标　　点	说　　明	标　　点	说　　明
:	冒号，具有多种应用功能	%	百分号，注释标记
;	分号，区分行及取消运行结果显示	!	惊叹号，调用操作系统运算
,	逗号，区分列及函数参数分隔符	=	等号，赋值标记
()	括号，指定运算的优先级	‘	单引号，字符串的标识符
[]	方括号，定义矩阵	.	小数点及对象域访问
{}	大括号，构造单元数组	…	续行符号

1.5 MATLAB R2018a 的文件管理

1.5.1 当前文件夹浏览器和路径管理器

工作文件夹窗口可显示或改变当前文件夹，还可以显示当前文件夹下的文件，以及提供文件搜索功能。与命令行窗口类似，该窗口也可以成为一个独立的窗口，如图 1-8 所示。

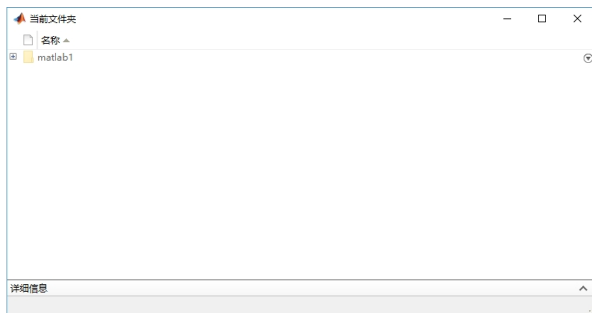


图 1-8 工作文件夹窗口

1.5.2 搜索路径及其设置

MATLAB 提供了专门的路径搜索器来搜索存储在内存中的 M 文件和其他相关文件，MATLAB 自带的文件所存放路径都被默认包含在搜索路径中，在 MATLAB 安装目录的“toolbox”文件夹中包含了所有此类目录和文件。

当用户在 MATLAB 提示符后输入一个字符串如“polyfit”后，MATLAB 进行的路径搜索步骤如下：

（1）检查 polyfit 是不是 MATLAB 工作区内的变量名，如果不是，执行下一步。

（2）检查 polyfit 是不是一个内置函数，如果不是，执行下一步。

（3）检查当前文件夹下是否存在一个名为 polyfit.m 的文件，如果没有，执行下一步。

（4）按顺序检查在所有 MATLAB 搜索路径中是否存在 polyfit.m 文件。

（5）如果仍然没有找到 polyfit，MATLAB 就会给出一条错误信息。

〇 提示

根据上述步骤可以推知，凡是不在搜索路径上的内容（文件和文件夹），都不能被 MATLAB 搜索到；当某一文件夹的父文件夹在搜索路径中而其本身不在搜索路径中时，则此文件夹并不会被搜索到。

一般情况下，MATLAB 系统的函数，包括工具箱函数，都是在系统默认的搜索路径之中的，但是用户设计的函数有可能没有被保存到搜索路径下，很多情况容易造成 MATLAB 误认为该函数不存在。因此，只要把程序所在的目录扩展成 MATLAB 的搜索路径即可。

下面将介绍 MATLAB 搜索路径的查看和设置方法。

1. 查看 MATLAB 的搜索路径

单击 MATLAB 主界面菜单/工具栏中的“设置路径”按钮，打开“设置路径”对话框，如图 1-9 所示。

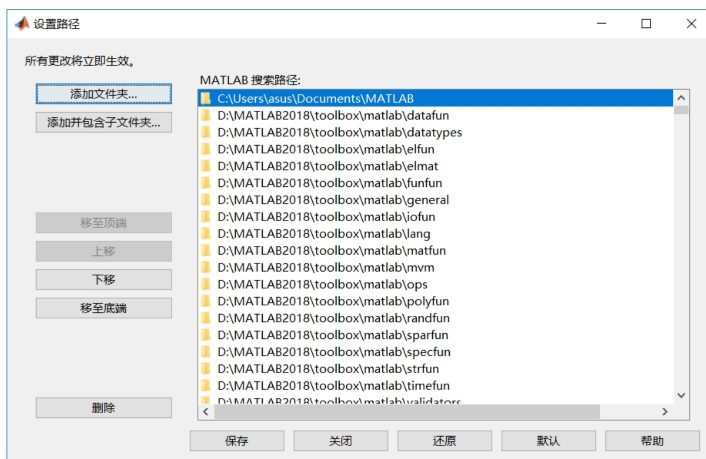


图 1-9 “设置路径”对话框

该对话框分为左右两部分，左侧几个按钮用来添加目录到搜索路径，还可从当前的搜索路径中移除选择的目录；右侧列表框列出了已经被 MATLAB 添加到搜索路径的目录。

此外，在命令行窗口中输入命令：

```
path
```

MATLAB 将会把所有的搜索路径列出来：

```
MATLABPATH


D:\用户目录\我的文档\MATLAB
    D:\ProgramFiles\MATLAB\R2018a\toolbox\matlab\datafun
    D:\ProgramFiles\MATLAB\R2018a\toolbox\matlab\datatypes
    D:\ProgramFiles\MATLAB\R2018a\toolbox\matlab\elfun
    D:\ProgramFiles\MATLAB\R2018a\toolbox\matlab\elmat
    D:\ProgramFiles\MATLAB\R2018a\toolbox\matlab\funfun
    D:\ProgramFiles\MATLAB\R2018a\toolbox\matlab\general
…
```

2. 设置 MATLAB 的搜索路径

MATLAB 提供了 3 种方法来设置搜索路径。

（1）在命令行窗口中输入：

```
edit path
```

或者：

```
 pathtool
```

或者通过 MATLAB 主界面菜单栏上的 File→Set Path 菜单，进入 Set Path 对话框，然后通过该对话框编辑搜索路径。

（2）在命令行窗口中输入：

```
path(path, 'path')        % 'path'是待添加的目录的完整路径
```

（3）在命令行窗口中输入：

```
addpath 'path' –begin% 'path'是待添加的目录的路径，将新目录添加到搜索路径的开始
addpath 'path' -end      % 'path'是待添加的目录的路径，将新目录添加到搜索路径的末端
```

1.6　MATLAB R2018a 的帮助系统

帮助文档是应用软件的重要组成部分，文档编制的质量直接关系到应用软件的记录、控制、维护、交流等一系列工作。

在当今软件生产中，没有一流的软件文档，就不会有一流的软件产品。MATLAB 提供的帮助系统，不仅对初学者，而且对熟练操作 MATLAB 的用户都有很大的帮助。

1.6.1　纯文本帮助

MATLAB 中的各个函数，不管是内建函数、M 文件函数，还是 MEX 文件函数等，一般都有 M 文件的使用帮助和函数功能说明，各个工具箱通常情况下也具有一个与工具箱名称相同的 M 文件来说明工具箱的构成内容。

因此，在 MATLAB 命令行窗口中，可以通过一些命令来获取这些纯文本的帮助信息。这些命令包括 help、lookfor、which、doc、get、type 等。

help 命令的常用调用方式为：

```
help FUN
```

执行该命令可以查询到有关于 FUN 函数的使用信息。例如要了解 sin 函数的使用方法，可以在命令行窗口中输入如下代码：

```
help sin
sin   Sine of argument in radians.
   sin(X) is the sine of the elements of X.

   See also asin, sind.

   sin 的参考页
   名为 sin 的其他函数
```

显示的帮助文档介绍了 sin 函数的主要功能、调用格式及相关函数的链接。

lookfor 命令的常用调用方式为：

```
lookfor topic
lookfor topic –all
```

执行该命令可以按照指定的关键字查找所有相关的 M 文件。例如：

```
lookfor inverse
ifft                - Inverse discrete Fourier transform.
ifft2                   - Two-dimensional inverse discrete Fourier transform.
ifftn               - N-dimensional inverse discrete Fourier transform.
ifftshift               - Inverse FFT shift.
acos                - Inverse cosine, result in radians.
acosd               - Inverse cosine, result in degrees.
acosh               - Inverse hyperbolic cosine.
```

```
    acot                    - Inverse cotangent, result in radian.
    acotd                   - Inverse cotangent, result in degrees.
   …
```

1.6.2 演示（Demos）帮助

通过 Demos 演示帮助，用户可以更加直观、快速地学习 MATLAB 中许多实用的知识。可以通过以下两种方式打开演示帮助。

选择 MATLAB 主界面菜单栏上的帮助下的示例命令。

在命令行窗口中输入：

```
demos
```

无论采用上述何种方式，执行命令后会都弹出帮助窗口，如图 1-10 所示。MATLAB Examples 里面又分为“Getting Started”“Mathematics”“Graphics”等一系列的演示。

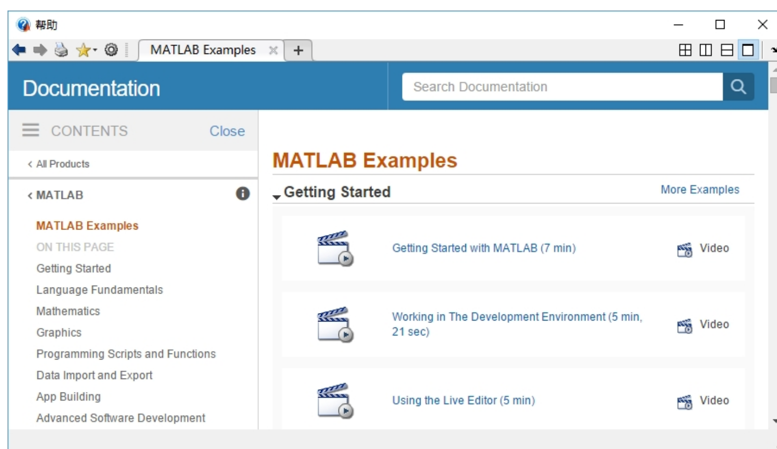


图 1-10 帮助窗口

1.6.3 帮助导航浏览器

帮助导航浏览器是 MATLAB 专门提供的一个独立的帮助子系统。该系统包含的所有帮助文件都存储在 MATLAB 安装目录下的 help 子目录下。用户可以采用以下两种方法打开帮助导航浏览器，分别为：

```
helpbrowser
```

或者：

```
doc
```

1.7 初步使用 MATLAB

下面以一个简单的示例向读者展示如何使用 MATLAB 进行简单的数值计算。

（1）双击桌面上的 MATLAB 图标，进入 MATLAB 的工作环境界面。

（2）在命令行窗口中输入“w=1/6*pi”，按 Enter 键，可以在工作区窗口看到变量 w，大小为 0.5236，命令行窗口中显示代码：

```
w =
   0.5236
```

（3）在命令行窗口中输入“y= sin(w*2/3)”，按 Enter 键，可以在工作区窗口看到变量 y，大小为 0.3420，命令行窗口中显示代码：

```
y =
   0.3420
```

（4）在命令行窗口中输入“z=sin(2*w)”，按 Enter 键，可以在工作区窗口看到变量 z，大小为 0.8660，命令行窗口中显示代码：

```
z =
   0.8660
```

提示

当命令后面有分号（半角符号格式）时，按 Enter 键后，命令行窗口中不显示运算结果；如果无分号，则在命令行窗口中显示运算结果。当希望先输入多条语句，然后再同时执行它们时，则在输入下一条命令时，要在按住 Shift 键的同时按 Enter 键进行换行输入。例如，比较使用“;”和不使用“;”的区别。

在命令行窗口中输入：

```
x=rand(2,3);
y=rand(2,3)
A=sin(x)
B=sin(2*y)
```

命令行窗口中依次输出结果：

```
y =
   0.3786   0.5328   0.9390
   0.8116   0.3507   0.8759
A =
   0.3016   0.4889   0.7137
   0.4869   0.7295   0.6007
B =
   0.6869   0.8751   0.9532
0.9986   0.6453   0.9836
```

1.8　本章小结

本章介绍了 MATLAB R2018a 的基本内容，主要包括 MATLAB 的发展历程、MATLAB R2018a 的工作环境和帮助系统。这些内容是使用 MATLAB 进行工作的基础，需要熟悉。但笔者不建议花费太多的时间进行专门的学习，只需要简单浏览即可，待到以后的学习和使用中需要时再根据需要进行学习。

第 2 章

MATLAB 基础知识

知识要点

MATLAB 是一个大型运算平台，参与运算的对象有数据流、信号流、逻辑关系及展示。如同计算器一样，在 MATLAB 中数学式的计算是直截了当的。但要了解这个大型计算器的使用方法并合理使用它，就先要了解一些 MATLAB 的基础知识。本章是整个 MATLAB 学习的基础，主要内容包括 MATLAB 软件平台上的各种数据类型、矩阵的基本运算、运算符及字符串数据的操作等。

学习要求	学习目标 知识点	了解	理解	应用	实践
	数据类型			√	
	创建各类型数据			√	√
	各种类型数据操作			√	√
	运算符		√		√
	字符串处理	√			√



2.1 数据类型

MATLAB 中的数据类型主要包括数值类型、逻辑类型、字符串、函数句柄、结构体和单元数组类型。这 6 种基本的数据类型都是按照数组形式存储和操作的。另外，MATLAB 中还有两种用于高级交叉编程的数据类型，分别是用户自定义的面向对象的用户类类型和 Java 类类型。

2.1.1 数值类型

基本的数值类型主要有整数、单精度浮点数和双精度浮点数，如表 2-1 所示。

表 2-1 数值类型数据的分类

数据格式	示 例	说 明
int8, uint8 int16, uint16 int32, uint32 int64, uint64	int32(820)	有符号和无符号的整数类型 相同数值的整数类型占用比浮点类型更少的内存 除 int64 和 uint64 类型外的所有整数类型，都可以进行数学运算
single	single(128.1)	单精度浮点类型 相同数值的单精度浮点类型比双精度浮点类型占用更少内存 精度与能够表示的数值范围比双精度浮点类型小
double	333.77 1.000-1.000i	双精度浮点类型，MATLAB 中默认的数值类型

MATLAB 中数值类型的数据包括有符号和无符号整数、单精度浮点数和双精度浮点数。在未加说明与特殊定义时，MATLAB 对所有数值按照双精度浮点数类型进行存储和操作。

在需要时，可以指定系统按照整数型或单精度浮点型对指定的数字或数组进行存储、运算等操作。相对于双精度浮点数格式，整数型与单精度浮点型的优点在于节省变量占用的内存空间，当然首先要在满足精度要求的前提下。

> ○ 提示
>
> MATLAB 会自动进行记忆体的使用和回收，而不像 C 语言，必须由使用者一一指定。这些功能使得 MATLAB 易学易用，使用者可专心致力于撰写程序。

1. 整数类型

MATLAB 中提供了 8 种内置的整数类型，这 8 种类型的存储占用位数、能表示的数值范围和转换函数均不相同，如表 2-2 所示。

不同的整数类型所占用的位数不同，因此能够表示的数值范围也不同。在实际应用中，应根据实际需要合理选择合适的整数类型。

由于 MATLAB 中数值的默认存储类型是双精度浮点类型，因此将变量设置为整数类型时，需要使用相应的转换函数，将双精度浮点数转换为指定的整数类型。

表 2-2　MATLAB 中的整数类型

整数类型	数值范围	转换函数
有符号 8 位整数	$-2^7 \sim 2^7-1$	int8
无符号 8 位整数	$0 \sim 2^8-1$	uint8
有符号 16 位整数	$-2^{15} \sim 2^{15}-1$	int16
无符号 16 位整数	$0 \sim 2^{16}-1$	uint16
有符号 32 位整数	$-2^{31} \sim 2^{31}-1$	int32
无符号 32 位整数	$0 \sim 2^{32}-1$	uint32
有符号 64 位整数	$-2^{63} \sim 2^{63}-1$	int64
无符号 64 位整数	$0 \sim 2^{64}-1$	uint64

在转换过程中，MATLAB 默认将待转换数值转换为与之最为接近的整数值，若小数部分为 0.5，则转换后的结果为与该浮点数最接近的两个整数中绝对值较大的一个。

另外，这些转换函数也可以将其他数据类型转换为指定的数据类型。在不超出数值范围的情况下，任意两个整数类型之间也可以通过转换函数进行相互转换。同时，由于不同的整数类型能够表示的数值范围不同，因此当运算结果超出相应的整数类型能够表示的范围时，就会出现一处错误，运算结果被置为该整数类型能够表示的最大值或最小值。

MATLAB 中还包含了几类不同运算法则的取整函数，也可以把浮点数转换成整数。这些取整函数及相应的转换方式如表 2-3 所示。

表 2-3　MATLAB 中的取整函数

函　　数	运算法则	示　　例
floor(x)	向下取整	floor (1.2)=1 floor (2.5)=2 floor (－2.5)= －3
ceil(x)	向上取整	ceil (1.2)=2 ceil (2.5)=3 ceil (－2.5)=－2
round(x)	取最接近的整数 如果小数部分是 0.5，则向绝对值大的方向取整	round (1.2)=1 round (2.5)=3 round (－2.5)=－3
fix(x)	向 0 取整	fix(1.2)=1 fix(2.5)=2 fix(－2.5)=－2

2．浮点数类型

MATLAB 中提供了单精度浮点数类型和双精度浮点数类型，其存储位宽、能够表示的数值范围、数值精度各方面均不相同，具体如表 2-4 所示。

由表 2-4 可知，单精度浮点类型的占用位数少，因此占用内存小，但能够表示的数值范围和数值的精度都比双精度浮点类型小。

由于 MATLAB 中的默认数值类型为双精度浮点类型，因此与创建整数类型数值一样，

也可以通过转换函数来实现创建单精度浮点类型。

表 2-4　MATLAB 中的浮点数类型

浮点类型	存储位宽	各数位的意义	数值范围	转换函数
单精度	32	0~22 位表示小数部分 23~30 位表示指数部分 31 位表示符号（0 正 1 负）	－3.40282e+038 ~－1.17549e-038 1.17549e-038 ~ 3.40282e+038	single
双精度	64	0~51 位表示小数部分 52~62 位表示指数部分 63 位表示符号（0 正 1 负）	－1.79769e+308 ~－2.22507e-308 2.22507e－308 ~ 1.79769e+308	double

双精度浮点数参与运算时，返回值的类型依赖于参与运算的其他数据类型。参与运算的其他数据为逻辑型、字符型时，返回结果为双精度浮点型；参与运算的其他数据为整数型时，返回结果为相应的整数类型；参与运算的其他数据为单精度浮点型时，返回结果为相应的单精度浮点型。

〇 提示

在 MATLAB 中，单精度浮点类型不能与整数类型进行算术运算。

例 2-1，浮点数参与的运算。

在命令行窗口中输入：

```
a=uint32(120);b=single(22.809);c=73.226;
ab=a*b
```

输出结果：

```
错误使用  .*
整数只能与相同类的整数或标量双精度值组合使用。
```

在命令行窗口中输入：

```
ac=a*c
```

输出结果：

```
ac =   8787
```

在命令行窗口中输入：

```
bc=b*c
```

输出结果：

```
bc = 1.6702e+003
```

在命令行窗口中输入：

```
str='hello'
```

输出结果：

```
str =
hello
```

在命令行窗口中输入：

```
newstr=str-44.3
```

输出结果：

```
newstr =
  59.7000   56.7000   63.7000   63.7000   66.7000
```

在命令行窗口中输入：

```
whos
```

输出结果：

```
  Name            Size           Bytes   Class       Attributes
  a               1x1            4       uint32
  ac              1x1            4       uint32
  b               1x1            4       single
  bc              1x1            4       single
c                 1x1            8       double
newstr            1x5            40      double
str               1x5            10      char
```

由于浮点数只占用一定的存储位宽，其中只有有限位分别用来存储指数部分和小数部分。因此，浮点类型能够表示的实际数值是有限且离散的，任何两个最近相邻的浮点数之间都有微小间隙，而处在间隙中的数值都只能用这两个相邻的浮点数之中的一个来表示。MATLAB 中提供了 eps 函数，可以获取一个数值和最接近该数值的浮点数之间的间隙。

例 2-2，浮点数的精度。

在命令行窗口中输入：

```
format long
eps(3)
```

输出结果：

```
ans =
   4.440892098500626e-016
```

在命令行窗口中输入：

```
eps(single(3))
```

输出结果：

```
ans =
  2.3841858e-007
```

3．复数

复数包括实部和虚部两部分。MATLAB 中默认使用字符 *i* 或 *j* 作为虚部标志。创建复数时，可以直接按照复数形式进行输入或者利用 complex 函数。MATLAB 库函数中关于复数的相关函数如表 2-5 所示。

表 2-5　MATLAB 中关于复数的相关函数

函　数	说　明	函　数	说　明
real(*z*)	返回复数 *z* 的实部	imag(*z*)	返回复数 *z* 的虚部
abs(*z*)	返回复数 *z* 的模	angle(*z*)	返回复数 *z* 的辐角
conj(*z*)	返回复数 *z* 的共轭复数	complex(*a*,*b*)	以 *a* 为实部、*b* 为虚部创建复数

4．无穷量（Inf）和非数值量（NaN）

MATLAB 中使用 Inf 和-Inf 分别代表正无穷量和负无穷量，NaN 表示非数值量。正负无穷量的产生一般是由于运算溢出，产生了超出双精度浮点数数值范围的结果，非数值量则是由于 0/0 或 Inf/Inf 类型的非正常运算而产生的，这两个 NaN 彼此是不相等的。

除了异常运算结果外，MATLAB 还提供了特定函数 Inf 和 NaN 来创建指定数值类型的无穷量和非数值量，生成结果默认为双精度浮点类型中还有一种特殊的指数类型的数据叫作非数，通常表示运算得到的数值结果超出了运算范围。非数的实部用 NaN 表示，虚部用 InF 表示。

例 2-3，无穷量及非数值量的产生和性质。

在命令行窗口中输入：

```
a = 0 / 0, b = log( 0 ),c = inf - inf
```

输出结果：

```
Warning: Divide by zero.
a = NaN
Warning: Log of zero.
b = -Inf
c = NaN
```

2.1.2　逻辑类型

逻辑类型的数据是指布尔类型的数据及数据之间的逻辑关系。除了传统的数学运算，MATLAB 还支持关系和逻辑运算。这些运算的目的是提供求解真/假命题的答案。

作为所有关系和逻辑表达式的输入，MATLAB 把任何非零数值当作真，把零当作假。所有关系和逻辑表达式的输出：对于真，输出为 1；对于假，输出为 0。

逻辑类型数据进行运算时需要用到关系操作符和逻辑运算符。

MATLAB 关系操作符如表 2-6 所示。

表 2-6　MATLAB 中的关系操作符

关系操作符	说　明
<	小于
< =	小于或等于
>	大于
> =	大于或等于
= =	等于
~ =	不等于

MATLAB 关系操作符能用来比较两个同样大小的数组，或用来比较一个数组和一个标量。在后一种情况中，标量和数组中的每一个元素相比较，结果与数组大小一样。

例 2-4，判断两个数组之间的元素是否相等。

在命令行窗口中输入：

```
A = 1 : 9, B = 10 - A
```

输出结果：

```
A =
    1    2    3    4    5    6    7    8    9
B =
    9    8    7    6    5    4    3    2    1
```

在命令行窗口中输入：

```
TrueorFalse = ( A == B )        %判断 A 与 B 中的元素是否相等
```

输出结果：

```
TrueorFalse =
    0    0    0    0    1    0    0    0    0
```

提示

“=”和“==”在 MATLAB 中的意义是不同的。“==”是对等号两边的两个变量进行比较，当它们相等时返回 1，不相等时返回 0；而“=”则是被用来将运算的结果赋给一个变量。

逻辑运算符提供了一种组合或否定关系表达式，如表 2-7 所示。

表 2-7　MATLAB 逻辑运算符

逻辑运算符	说　明
&	与
|	或
~	非

例 2-5，判断两个数组之间的元素是否相等。

在命令行窗口中输入：

```
A = 1 : 9;
 TrueorFalse = ( A > 2 ) & ( A < 6 )        %判断 A 中元素是否在 2～6 之间
```

输出结果：

```
TrueorFalse =
    0    0    1    1    1    0    0    0    0
```

除了上述关系与逻辑运算符，MATLAB 还提供了大量其他关系与逻辑函数。

xor(x , y)指令的功能为异或运算，x 和 y 同为零（假）或非零（真）时返回 0，否则返回 1。

any(x)指令的功能为判断是否为零向量或零矩阵（即向量或矩阵中的元素全部为零），如果是零向量或零矩阵，则返回 1，否则返回 0。

除此之外，MATLAB 还提供了大量的函数，在运算过程中用来测试特殊值或条件是否存在，并返回相应的表示结果的逻辑值，如表 2-8 所示。

表 2-8　测试函数

函数名称	函数功能
finite	元素有限，返回真值
Isempty	参量为空，返回真值
isglobal	参量是一个全局变量，返回真值
ishold	当前绘图保持状态是“ON”，返回真值
isieee	计算机执行 IEEE 算术运算，返回真值
isinf	元素无穷大，返回真值
isletter	元素为字母，返回真值
isnan	元素为不定值，返回真值
isreal	参量无虚部，返回真值
isspace	元素为空格字符，返回真值
isstr	参量为一个字符串，返回真值
isstudent	MATLAB 为学生版，返回真值
isunix	计算机为 UNIX 系统，返回真值

2.1.3　字符和字符串

在 MATLAB 中，文本当作特征字符串或简单地当作字符串。字符串能够显示在屏幕上，也可以用来构成一些命令，这些命令在其他的命令中用于求值或者被执行。

在 MATLAB 中可能会遇到对字符和字符串的操作。一个字符串是存储在一个行向量中的文本，这个行向量中的每一个元素代表一个字符。实际上，元素中存放的是字符的内部代码，即 ASCII 码。

当在屏幕上显示字符变量的值时，显示出来的是文本，而不是 ASCII 数字。由于字符串是以向量的形式来存储的，因此可以通过它的下标对字符串中的任何一个元素进行访问。字符矩阵也可以通过下标索引进行访问，但是矩阵的每行字符数必须相同。

字符串一般是 ASCII 值的数值数组，它作为字符串表达式进行显示。

例 2-6，字符串属性示例。

在命令行窗口中输入：

```
String ='Every good boy does fun.';
size(String)
```

输出结果：

```
ans =
    1    24
```

在命令行窗口中输入：

```
whos
```

输出结果：

```
Name            Size                Bytes    Class    Attributes
```

```
ans      1x2             16      double
String       1x24              48       char
```

一个字符串是由单引号括起来的简单文本。在字符串里的每个字符是数组里的一个元素，字符串的存储要求每个字符 8 字节，如同 MATLAB 的其他变量。

因为 ASCII 字符只要求 1 字节，故这种存储要求是浪费的，7/8 所分配的存储空间无用。然而，对字符串保持同样的数据结构可以简化 MATLAB 的内部数据结构。所给出的字符串操作并不是 MATLAB 的基本特点，但这种表达是方便和可接受的。

为了了解下面字符串的 ASCII 表达，只需对字符串执行一些算术运算。最简单和计算上最有效的方法是取数组的绝对值。

例 2-7，字符串的 ASCII 表达。

在命令行窗口中输入：

```
String ='Every good boy does fun.';
U=abs (String)
```

输出结果：

```
U =
  Columns 1 through 12
69   118   101   114   121   32   103   111   111   100   32   98
  Columns 13 through 24
 111   121   32   100   111   101   115   32   102   117   110   46
```

在命令行窗口中输入：

```
U=U+0
```

输出结果：

```
U =
  Columns 1 through 12
69   118   101   114   121   32   103   111   111   100   32   98
  Columns 13 through 24
 111   121   32   100   111   101   115   32   102   117   110   46
```

上例中，加零到字符串并没有改变它的 ASCII 表达。

因为字符串是数值数组，所以它们可以用 MATLAB 中所有可利用的数组操作工具进行操作。

例 2-8，字符串数组的索引示例。

在命令行窗口中输入：

```
String ='Every good boy does fun.';
U =String(7:10)
```

输出结果：

```
U =
    good
```

在命令行窗口中输入：

```
U =String(10:-1:7)
```

输出结果：

```
U =
```

```
    doog
```

上例中字符串像数组一样进行编址。这里元素 7～10 包含单词 good。

字符串内的单引号是由两个连续的单引号来表示的。

例 2-9，字符串中的单引号。

在命令行窗口中输入：

```
String ='It''s not the manual!'
```

输出结果：

```
String =
    It's not the manual!
```

字符串的连接可以直接将字符串数组连接来实现。

例 2-10，字符串的连接。

在命令行窗口中输入：

```
U ='Hello,';
V =' world!';
W = [U V]
```

输出结果：

```
W =
    Hello, world!
```

2.1.4　函数句柄

在 MATLAB 平台中，对函数的调用方法分为直接调用法和间接调用法。

- 直接调用函数，被调用的函数通常被称为子函数。但是子函数只能被与其 M 文件同名的主函数或在 M 文件中的其他函数所调用，同时在一个文件中只能有一个主函数。
- 而使用函数句柄对函数进行调用则避免了上述问题。函数句柄实际上提供了一种间接调用函数的方法。创建函数句柄需要用到操作符@。对 MATLAB 库函数中提供的各种 M 文件中的函数和使用者自主编写的程序中的内部函数，都可以创建函数句柄，从而可以通过函数句柄来实现对这些函数的间接调用。

创建函数句柄的一般句法格式为：

```
Function_Handle = @Function_Filename;
```

其中，

- Function_Filename 是函数所对应的 M 文件的名称或 MATLAB 内部函数的名称。
- @是句柄创建操作符。
- Function_Handle 变量保存了这一函数句柄，并在后续的运算中作为数据流进行传递。

例如，F_Handle = @cos 就创建了 MATLAB 内部函数 cos 的句柄，并将其保存在 F_Handle 变量中，后续的运算过程中就可以通过 F_Handle(*x*)来实现 cos(*x*)的功能。

通过函数句柄调用函数时，也需要指定函数的输入参数。例如可以通过 F_Handle(arg1, arg2,…,argn)这样的调用格式来调用具有多个输入参数的函数。对于那些没有输入参数的函

数，在使用句柄调用时，在句柄变量之后的圆括号中不填写变量名即可，即 F_Handle()。

例 2-11，函数句柄的创建与调用。

在命令行窗口中输入：

```
F_Handle = @cos
```

输出结果：

```
F_Handle =
   @cos
```

在命令行窗口中输入：

```
x = 0 : 0.25 * pi : 2 * pi;
F_Handle( x )            %通过函数句柄调用函数
```

输出结果：

```
ans =
  1.0000  0.7071  0.0000  -0.7071  -1.0000  -0.7071 -0.00000.70711.0000
```

MATLAB 库函数中提供了大量关于处理函数句柄的操作函数，将函数句柄的功能与其他数据类型联系起来，扩展了函数句柄的应用。函数句柄的简单操作函数如表 2-9 所示。

表 2-9 函数句柄的操作函数

函数名称	函数功能
functions(funhandle)	返回一个结构体，存储了函数的名称、函数类型（simple 或 overloaded），以及函数 M 文件的位置
func2str(funhandle)	将函数句柄转换为函数名称字符串
str2func(str)	将字符串代表的函数转换为函数句柄
save filename.mat funhandle	将函数句柄保存在*.mat 文件中
load filename.mat funhandle	把*.mat 文件中存储的函数句柄加载到工作区
isa(var, 'function_handle')	检测变量 var 是否是函数句柄
isequal(funhandlea, funhandleb)	检测两个函数句柄是否对应于同一个函数

例 2-12，函数句柄的基本操作。

在命令行窗口中输入：

```
F_Handlea=@exp
```

输出结果：

```
F_Handlea =
   @exp
```

在命令行窗口中输入：

```
F_Handleb=@log
```

输出结果：

```
F_Handleb =
   @log
```

在命令行窗口中输入：

```
functions(F_Handlea )
```

输出结果：

```
ans =
   function: 'exp'
       type: 'simple'
       file: ''
```

在命令行窗口中输入：

```
isa(F_Handlea, 'function_handle')        %判断 F_Handlea 是否是函数句柄
```

输出结果：

```
ans = 1
```

在命令行窗口中输入：

```
isequal(F_Handlea, F_Handleb)       %判断两个函数句柄是否对应同一函数
```

输出结果：

```
ans = 0
```

2.1.5　结构体类型

MATLAB 中的结构体与 C 语言中的结构体类似，一个结构体可以通过字段存储多个不同类型的数据。因此，结构体相当于一个数据容器，把多个相关联的不同类型的数据封装在一个结构体对象中。

如图 2-1 所示，结构体 student 中有 4 个字段，姓名字段 name 中存储了一个字符串类型的数据；年级字段 grade 中存储了一个浮点数值；科目字段 subject 中存储了一个一维字符串数组；成绩字段 result 中存储了一个一维数组。

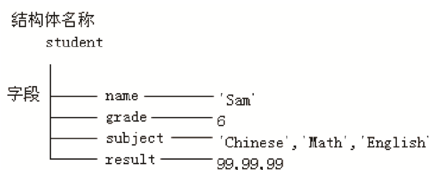


图 2-1　结构体 student 的示意图

一个结构体中可以具有多个字段，每个字段又可以存储不同类型的数据，通过这种方式就把多个不同类型的数据组织在一个结构体对象中。

创建结构体对象的方法有两种，可以直接通过赋值语句给结构体的字段赋值，也可以使用结构体创建函数 struct。两种方法的具体操作步骤如下：

（1）通过字段赋值创建结构体。在对结构体的字段进行赋值时，赋值表达式的变量名使用“结构体名称.字段名称”的形式书写，对同一个结构体可以进行多个字段的赋值。

例 2-13，通过赋值创建结构体。

在命令行窗口中输入：

```
Student.Name='Sam';
Student.Grade=6;
Student.Subject={'Chinese','Math','English'};
Student.Result={99,99,99};
```

```
 Student
```

输出结果：

```
Student =
     Name: 'Sam'
     Grade: 6
 Subject: {1×3 cell}
    Result: {[99]  [99]  [99]}
在命令行窗口中输入：
whos
```

输出结果：

```
  Name          Size          Bytes  Class          Attributes
  Student        1x1            930                  struct
```

上例中，通过对 4 个字段赋值，创建了结构体对象 Student，然后用 whos 函数分析出 Student 是一个 1×1 的结构体数组。

○ 注意

在进行字段赋值操作时，没有明确赋值的字段，MATLAB 默认赋值为空数组。通过圆括号索引进行字段赋值，还可以创建任意尺寸的结构体数组。需要注意的是，同一个结构体数组中的所有结构体对象具有相同的字段组合。

struct 函数的句法形式为：

```
StrArray = struct('field1', var1, 'field2', var2,…,'fieldn', varn)
```

上述语句可以创建结构对象 StrArray，并将其 *n* 个字段分别赋值为 var1,var2,⋯, varn。

例 2-14，利用 struct 函数创建结构体。

在命令行窗口中输入：

```
Schedule(2)=struct('Day','Thursday','Time','15:00','Number',18)
```

输出结果：

```
Schedule =
1×2 struct array with fields:
   Day
   Time
   Number
```

在命令行窗口中输入：

```
Schedule(1)       %结构体第一个元素没有赋值，因此所有字段均为空数组
```

输出结果：

```
ans =
      Day: []
     Time: []
   Number: []
```

在命令行窗口中输入：

```
ScheduleArray=repmat(struct('Day','Thursday','Time','15:00','Number',18),1,2)
```

输出结果：

```
ScheduleArray =
1×2 struct array with fields:
   Day
```

```
   Time
   Number
```

在命令行窗口中输入：

```
ScheduleArray(1)              %1×2 的结构体数组的两个元素完全相同
```

输出结果：

```
ans =
    Day: 'Thursday'
    Time: '15:00'
   Number: 18
```

在命令行窗口中输入：

```
ScheduleArray(2)
```

输出结果：

```
ans =
     Day: 'Thursday'
     Time: '15:00'
  Number: 18
```

在命令行窗口中输入：

```
newArray=struct('Day',{'Thursday','Friday'},'Time',{'15:00','9:00'},
'Number',{18,6})
```

输出结果：

```
newArray =
1×2 struct array with fields:
   Day
   Time
   Number
```

在命令行窗口中输入：

```
newArray(1)
```

输出结果：

```
ans =
      Day: 'Thursday'
     Time: '15:00'
   Number: 18
```

在命令行窗口中输入：

```
newArray(2)
```

输出结果：

```
ans =
         Day: 'Friday'
         Time: '9:00'
     Number: 6
```

2.1.6 数组类型

在 MATLAB 中进行运算的所有数据类型，都是按照数组及矩阵的形式进行存储和运算

的，而二者在 MATLAB 中的基本运算性质不同，阵列强调元素对元素的运算，而矩阵则采用线性代数的运算方式。本节主要介绍数组类型，关于矩阵的详细运算语法，可参看第 3 章的详细说明。

数组的属性及数组之间的逻辑关系，是编写程序时非常重要的两个方面。在 MATLAB 平台上，数组的定义是广义的，数组的元素可以是任意的数据类型，例如可以是数值、字符串、指针等。

利用数组的构建方法可以直接对变量进行赋值。

例 2-15，对变量赋值创建数组。

在命令行窗口中输入：

```
Array = [1 2 3 4 5 6]
```

输出结果：

```
Array =
    1    2    3    4    5    6
```

在 MATLAB 中可以使用冒号“:”来代表一系列数值，有时也使用它来定义数组。其句法格式如下：

```
Array = i : k
```

创建从 *i* 开始、步长为 1、到 *k* 结束的数字序列，即 I,*i*+1,*i*+2,⋯,*k*。如果 *i*>*k*，MATLAB 则返回一个空矩阵。数字 *i* 和 *k* 不必是整数，该序列的最后一个数小于或等于 *k*。

```
Array = i : j : k
```

创建从 *i* 开始、步长为 *j*、到 *k* 结束的数字序列，即 I,*i*+*j*,*i*+2*j*,⋯,*k*。如果 *j*= 0，则返回一个空矩阵。数字 *i*、*j* 和 *k* 不必是整数，该序列的最后一个数小于或等于 *k*。

还有一些预定义函数也可以用来创建线性序列和逻辑序列。

```
Array = linspace(a, b,100)
```

在区间[*a*, *b*]上创建一个有 100 个元素的向量，这 100 个数把整个区间线性分隔。

```
Array = linspace(a,b,n,100)
```

在区间[*a*, *b*]上创建一个有 *n* 个元素的向量。这个命令和冒号表示形式相近，但是它直接定义了数据的个数。

例 2-16，创建等差数列。

在命令行窗口中输入：

```
Array_a=0:5,Array_b=linspace(0,5,6)
```

输出结果：

```
Array_a =
    0    1    2    3    4    5
Array_b =
    0    1    2    3    4    5
```

当数组的元素个数为 0 时，就称数组为空。空数组是特殊的数组，它不含有任何元素。空数组主要用于逻辑运算、数组声明、数组的清空等。

例 2-17，创建空数组。

在命令行窗口中输入：

```
Array_Empty=[]
```

输出结果：

```
Array_Empty =
             []
```

2.1.7 单元数组类型

1. 概述

单元（Cell）数组是一种无所不包的广义矩阵。组成单元数组的每一个元素称为一个单元。每一个单元可以包括一个任意数组，如数值数组、字符串数组、结构体数组或另外一个单元数组，因而每一个单元可以具有不同的尺寸和内存占用空间。

> ○ 注意
>
> 和一般的数值数组一样，单元数组的维数不受限制，可以是一维、二维或多维。

MATLAB 中使用单元数组的目的在于，它可以把不同类型的数据归并到一个数组中。

> ○ 注意
>
> 单元数组的创建有两种方法：使用赋值语句创建单元数组和利用 cell 函数创建数组。

（1）使用赋值语句创建单元数组。与一般数组有所不同的是，单元数组使用花括号“{}”来创建，使用逗号“,”或空格来分隔每一个单元，使用分号“;”来分行。

例 2-18，创建单元数组。

在命令行窗口中输入：

```
C = {'x',[1;3;6];10,pi}
whos C
```

程序运行结果如下：

```
C =
    'x'     [3x1 double]
    [10]    [   3.1416]

  Name      Size          Bytes  Class    Attributes
  C        2x2            282  cell
```

（2）利用 cell 函数创建空单元数组。

cell 函数的调用格式如下：

```
cellName = cell（m, n）
```

该函数创建一个 $m \times n$ 的空单元数组，其每一个单元均为空矩阵。

例 2-19，创建空单元数组。

在命令行窗口中输入：

```
a = cell(2,2)
b = cell(1)
```

```
whos
```

程序运行结果如下：

```
a =
    []    []
    []    []
b =
    {[]}
  Name      Size           Bytes  Class    Attributes
  a         2x2               16  cell
  b         1x1                4  cell
```

同一般的数值数组一样，单元数组的内存空间也是动态分配的。因此，使用 cell 函数创建空单元数组的主要目的是为该单元数组预先分配连续的存储空间，以节约内存占用，提高执行效率。

2．单元数组的寻访

在单元数组中，单元和单元中的内容是两个不同范畴的东西，因此，寻访单元和单元中的内容是两个不同的操作。MATLAB 为上述两种操作设计了相对应的操作对象：单元外标识（Cell Indexing）和单元内编址（Content Addressing）。

对于单元数组 *C*，*C*(*m*,*n*)指的是单元数组中第 *m* 行第 *n* 列的单元，而 *C*{*m*,*n*}指的是单元数组中第 *m* 行第 *n* 列单元中的内容。

例 2-20，单元数组的寻访。

在命令行窗口依次输入命令：

```
C = {3,[4 7;6 6;80 9],'string';sin(pi/8),3>10,'code'}
unitVal_1 = C(2,2)
class(unitVal_1)
unitVal_2 = C{2,2}
class(unitVal_2)
```

依次得到的结果如下：

```
C =
    [              3]    [3x2 double]    'string'
    [0.382683432365090]    [       0]    'code'
unitVal_1 =  [0]
ans = cell
unitVal_2 = 0
ans = logical
```

3．单元数组的操作

单元数组的操作包括合并、删除单元数组中的指定单元、改变单元数组的形状等。

（1）单元数组的合并。

例 2-21，单元数组的合并。

在命令行窗口依次输入以下命令：

```
a{1,1}='cellclass';
a{1,2}=[1 2 2];
a{2,1}=['a','b','c'];
a{2,2}=[9 5 6];
a
b = {'Jan'}
```

```
c = {a b}
```

得到的结果如下：

```
a =
   'cellclass'   [1x3 double]
   'abc'        [1x3 double]
b =  'Jan'
c =  {2x2 cell}   {1x1 cell}
```

（2）单元数组中指定单元的删除。

如果要删除单元数组中指定的某个单元，只需要将空矩阵赋给该单元，即：

```
C{ m,n} = []
```

例 2-22，有一个单元数组 *C*，删除其中的单元数组。

在命令行窗口依次输入下列命令：

```
C = {ones(3),'Hello, World',zeros(5),[20,4,6]}
C{1,4} = []
```

得到的结果如下：

```
C =    [3x3 double]   'Hello, World'   [5x5 double]   [1x3 double]
C =    [3x3 double]   'Hello, World'   [5x5 double]   []
```

（3）使用 reshape 函数改变单元数组的形状。

reshape 函数的调用格式为：

```
trimC = reshape（C, M, N）
```

该函数将单元数组 *C* 改变成一个具有 *M* 行 *N* 列的新单元数组。

例 2-23，将例 2-22 中的单元数组 *C*（1×4）改变成 new*C*（4×1）。

在命令行窗口中输入：

```
newC = reshape(C,4,1)
```

得到的结果如下：

```
newC =
   [3x3 double]
   'Hello, World'
   [5x5 double]
   []
```

2.1.8　map 容器类型

1. map 容器类型及 map 类概述

map 的本意是映射，就是可以将一个量映射到另一个量。比如将一个字符串映射为一个数值，那个字符串就是 map 的键（key），那个值就是 map 的数据（value）。因此，可以将 map 容器理解为一种快速查找数据结构的键。

对一个 map 元素进行寻访的索引称为“键”。一个“键”可以是以下任何一种数据类型：

- 1×*N* 字符串。
- 单精度或双精度实数标量。
- 有符号或无符号标量整数。

这些键和其对应的数据存储在 map 中。一个 map 的每一个条目都包括唯一的键和相对应的数据。map 中存储的数据可以是任何类型，包括数值类型、字符或字符串类型、结构体类型、单元类型或其他 map。

一个 map 是 MATLAB 类的一个对象。map 类的所有对象具有 3 种属性，如表 2-10 所示。用户不能直接对这些属性进行修改，但可以通过作用于 map 类的函数进行修改。

表 2-10　map 类的属性

属　性	说　明	默 认 值
Count	无符号 64 位整数，表示 map 对象中存储的 key/value 对的总数	0
KeyType	字符串，表示 map 对象中包括的 key 的类型	char
ValueType	字符串，表示 map 对象中包括的数据类型	any

map 属性的查看方法为 map 名＋小数点“.”＋map 的属性名。例如，为了查看 mapW 对象包括的数据类型，需要使用 mapW.ValueType。

2．创建 map 对象

map 是一个 map 类中的对象，由 MATLAB 中名为“容器”的一个包来定义，可以通过构造函数来创建，其创建方法如下：

```
mapObj = containers.Map({key1,key2,…},{val1,val2,…})
```

当键和值是字符串时，需要对上述语法稍作变更，即：

```
mapObj = containers.Map({'key1','key2',…},{val1,val2,…})
```

例 2-24，创建一个名为 schedulemap 的 map 对象来存储如表 2-11 所示的课程表。

表 2-11　课程表

星 期 一	星 期 二	星 期 三	星 期 四	星 期 五
数学	语文	历史	地理	生物

创建过程如下：

```
schedulemap = containers.Map({'Monday','Tuesday','Wednesday','Thursday',
'Friday'},
{'Maths','Chinese','History','Geography','Biology'})
```

得到的结果如下：

```
schedulemap =
  Map (带属性):
        Count: 5
      KeyType: char
    ValueType: char
```

此外，map 对象的创建可以分为两个步骤：首先创建一个空 map 对象；然后使用 keys 和 values 方法对其内容进行补充。空 map 对象的创建方法如下：

```
newMap = containers.Map()
```

得到的结果如下：

```
newMap =
  Map (带属性):
```

```
       Count: 0
     KeyType: char
   ValueType: any
```

3．查看/读取 map 对象

（1）查看 map 对象。

map 对象中的每个条目包括两个部分：一个唯一的键及其对应的值。可以通过使用 keys 函数查看 map 对象中包含的所有键；通过 values 函数查看所有的值。

例 2-25，查看例 2-24 中创建的 map 对象。

在命令行窗口依次输入：

```
keys(schedulemap)
values(schedulemap)
```

得到的结果如下：

```
ans =     'Friday'   'Monday'   'Thursday'   'Tuesday'   'Wednesday'
ans =     'Biology'   'Maths'   'Geography'   'Chinese'   'History'
```

（2）读取 map 对象。

在创建好一个 map 对象后，用户可以对其进行数据的寻访。寻访指定键（keyName）所对应的值（valueName）使用的格式如下：

```
valueName = mapName（keyName）
```

当键名是一个字符串时，需使用单引号将键名括起来。

例 2-26，通过使用键名访问例 2-24 中创建的 schedulemap 对象中的内容。

输入内容如下：

```
course = schedulemap('Wednesday')
```

得到的结果如下：

```
course =History
```

如果需要对多个键进行访问，可以使用 values 函数，如：

```
values(schedulemap,{'Monday','Thursday'})
```

得到的结果如下：

```
ans =   'Maths'   'Geography'
```

> **提示**
>
> 在对多个键进行访问时，不能像在其他数据类型中那样使用冒号“:”，这将导致错误产生。例如输入：
>
> ```
> schedulemap('Monday':'Thursday')
> ```
>
> 结果弹出如下错误信息：
>
> 警告：冒号操作数必须为实数标量。

4．编辑 map 对象

（1）从 map 对象中删除 keys/values 对。

用户可以使用 remove 函数从 map 对象中删除 keys/values 对，该函数的调用格式为：

```
remove('mapName', 'keyName')
```

式中，mapName 和 keyName 分别为 map 对象名称及需要删除的键名。执行该命令后，

MATLAB 系统删除指定的键名及其相对应的值。

例 2-27，删除 schedulemap 中的“星期四”及其对应的科目。

在命令行窗口依次输入：

```
schedulemap = containers.Map({'Monday','Tuesday','Wednesday','Thursday',
      'Friday'},{'Maths','Chinese','History','Geography','Biology'})
newMap = containers.Map()
remove(schedulemap,'Thursday')
keys(schedulemap)
values(schedulemap)
```

得到的结果如下：

```
ans =
   Map (带属性):
      Count: 4
    KeyType: char
   ValueType: char
ans =    'Friday'    'Monday'    'Tuesday'    'Wednesday'
ans =    'Biology'    'Maths'    'Chinese'    'History'
```

（2）添加 keys/values 对象。

当用户向一个 map 对象中写入新元素的值时，需要提供键名，而且该键的类型必须和 map 中的其他键一致。该操作的调用格式为：

```
existingMapObj(newKeyName)= newValue
```

例 2-28，为 schedulemap 添加“星期六”及其对应的科目“public elective course”。

在命令行窗口依次输入：

```
schedulemap('Saturday') = 'public elective course'
keys(schedulemap)
values(schedulemap)
```

得到的结果如下：

```
schedulemap =
  Map (带属性):
      Count: 6
    KeyType: char
   ValueType: char
ans =    'Friday'    'Monday'    'Saturday'    'Thursday'    'Tuesday'
  'Wednesday'
ans =    'Biology'    'Maths'    [1x22 char]    'Geography'    'Chinese'
  'History'
```

（3）修改 keys。

如果需要在保持值不变的情况下对键名进行更改，首先要删除键名和对应的值，然后再添加一个有正确键名的新条目。

例 2-29，根据例 2-28 中的 schedulemap 结果，修改“星期六”及其对应的科目“公共选修课”为“星期天”及其对应的科目“MBA”。

在命令行窗口依次输入：

```
remove(schedulemap,'Saturday');
schedulemap('Sunday') = 'MBA';
keys(schedulemap)
```

```
values(schedulemap)
```

程序运行结果如下：

```
ans =    'Friday'    'Monday'    'Sunday'    'Thursday'    'Tuesday'    'Wednesday'
ans =     'Biology'    'Maths'    'MBA'    'Geography'    'Chinese'    'History'
```

（4）修改 values。

通过赋值操作，覆盖原有的值，即可对 map 对象中的值进行修改。

例 2-30，修改“星期一”的科目为“英语”。

在命令行窗口依次输入：

```
schedulemap('Monday')
schedulemap('Monday') = 'english';
keys(schedulemap)
values(schedulemap)
```

程序运行结果如下：

```
ans = Maths
ans = 'Friday'    'Monday'    'Saturday'    'Thursday'    'Tuesday'    'Wednesday'
ans = 'Biology'    'english'    'MBA'    'Geography'    'Chinese'    'History'
```

2.2 基本矩阵操作

2.2.1 矩阵和数组的概念及其区别

对矩阵的基本操作，主要有矩阵的构建、矩阵维度与矩阵大小的改变、矩阵的索引、矩阵属性信息的获取、矩阵结构的改变等。对于这些操作，MATLAB 中都有固定指令或相应的库函数与之相对应。

在数学上，定义由 $m \times n$ 个数 $a_{ij}\left(i=1,2,\cdots,m;\ j=1,2,\cdots,n\right)$ 排成的 m 行 n 列的数表

$$\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

为 m 行 n 列矩阵，并用大写黑体字母表示它。

只有一行的矩阵

$$A=\left(a_1 \quad a_2 \quad \cdots \quad a_n\right)$$

称为行向量。

同理，只有一列的矩阵

$$A=\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

称为列向量。

矩阵最早来自于方程组的系数及常数所构成的方阵，这一概念由 19 世纪英国数学家凯利首先提出。

数组是在程序设计中，为了处理方便，把具有相同类型的若干变量按有序的形式组织起来的一种形式。这些按序排列的同类数据元素的集合称为数组。

在 MATLAB 中，一个数组可以分解为多个数组元素，这些数组元素可以是基本数据类型或构造类型。因此按数组元素的类型不同，数组又可分为数值数组、字符数组、单元数组、结构数组等各种类别。

由此可见，矩阵和数组在 MATLAB 中存在很多方面的区别，主要有以下几个方面：

- 矩阵是数学上的概念，而数组是计算机程序设计领域的概念。
- 作为一种变换或者映射算符的体现，矩阵运算有着明确而严格的数学规则。而数组运算是 MATLAB 软件定义的规则，其目的是为了使数据管理方便，操作简单，命令形式自然，执行计算有效。

两者间的联系主要体现在：在 MATLAB 中，矩阵是以数组的形式存在的。因此，一维数组相当于向量，二维数组相当于矩阵，所以矩阵是数组的子集。

2.2.2　矩阵的构造

矩阵的构建方式有两种，一种与单元数组相似，可以对变量直接进行赋值；另一种是 MATLAB 中提供构建特殊矩阵的指令，如表 2-12 所示。

表 2-12　特殊矩阵的构建函数

函数名称	函数功能
ones(*n*)	构建一个 *n*×*n* 的 1 矩阵（矩阵的元素全部是 1）
ones(*m* , *n* ,⋯, p)	构建一个 *m*×*n*×⋯×*p* 的 1 矩阵
ones(size(***A***))	构建一个和矩阵 ***A*** 同样大小的 1 矩阵
zeros(*n*)	构建一个 *n*×*n* 的 0 矩阵（输出矩阵的元素全部是 0）
zeros(*m* , *n* ,⋯, *p*)	构建一个 *m*×*n*×⋯×*p* 的 0 矩阵
zeros(size(***A***))	构建一个和矩阵 ***A*** 同样大小的 0 矩阵
eye(*n*)	构建一个 *n*×*n* 的单位矩阵
eye(*m*, *n*)	构建一个 *m*×*n* 的单位矩阵
eye(size(***A***))	构建一个和矩阵 ***A*** 同样大小的单位矩阵

（续表）

函数名称	函数功能
magic(*n*)	构建一个 *n*×*n* 的矩阵，其每一行、每一列的元素之和都相等
rand(*n*)	构建一个 *n*×*n* 的矩阵，其元素为 0～1 之间均匀分布的随机数
rand(*m* , *n* ,⋯, *p*)	构建一个 *m*×*n*×⋯×*p* 的矩阵，其元素为 0～1 之间均匀分布的随机数
randn (*n*)	构建一个 *n*×*n* 的矩阵，其元素为零均值、单位方差的正态分布随机数
randn (*m* , *n* ,⋯, *p*)	构建一个 *m*×*n*×⋯×*p* 的矩阵，其元素为零均值、单位方差的正态分布随机数
diag(***x***)	构建一个 *n* 维的方阵，它的主对角线元素值取自向量 *x*， 其余元素的值都为 0
diag(***A*** , *k*)	构建一个由矩阵 ***A*** 第 *k* 条对角线的元素组成的列向量 *k*= 0 为主对角线；*k*< 0 为下第 *k* 条对角线；*k*> 0 为上第 *k* 条对角线
diag(***x*** , *k*)	构建一个(*n*+|*k*|)×(*n*+|*k*|)维的矩阵，该矩阵的第 *k* 条对角线元素取自向量 ***x***，其余元素都为 0（关于参数 *k*，参考上个命令）
triu(***A***)	构建一个和 ***A*** 大小相同的上三角矩阵，该矩阵的主对角线上元素为 ***A*** 中相应元素，其余元素都为 0
triu(***A*** , *k*)	构建一个和 ***A*** 大小相同的上三角矩阵，该矩阵的第 *k* 条对角线及其以上元素为 ***A*** 中相应元素，其余元素都为 0
tril(***A***)	构建一个和 ***A*** 大小相同的下三角矩阵，该矩阵的主对角线上元素为 ***A*** 中相应元素，其余元素都为 0
tril(***A***, *k*)	构建一个和 ***A*** 大小相同的下三角矩阵，该矩阵的第 *k* 条对角线上及其以下元素为 ***A*** 中相应元素，其余元素都为 0

1．建立简单矩阵

简单矩阵采用矩阵构造符号——方括号“[]”，将矩阵元素置于方括号内，同行元素之间用空格或逗号隔开，行与行之间用分号“;”隔开，格式如下：

```
matrixName = [element11,element12,element13;element21,element22,element23]
matrixName = [element11 element12 element13;element21 element22 element23]
```

例 2-31，简单矩阵构造示例。分别构造一个二维矩阵、一个行向量、一个列向量。

在命令行窗口依次输入下列命令：

```
A = [2,3,5;3,6,10]     % 使用逗号和分号构造二维矩阵
B = [2 3 5;3 6 10]     % 使用空格和分号构造二维矩阵
V1 = [8,59,60,33]      % 构造行向量
V2 = [5;8;3;4;9]       % 构造列向量
```

程序运行过程中的输出如下：

```
A =  2     3     5
     3     6    10
B =  2     3     5
     3     6    10
V1 =    8    59    60    33
V2 =   5
     8
     3
     4
     9
```

2．建立特殊矩阵

特殊矩阵是指非零元素或零元素的分布有一定规律的矩阵，常见的有对称矩阵、三角矩阵和对角矩阵等。

例 2-32，特殊矩阵构造示例。

在命令行窗口中输入：

```
OnesMatrix = ones( 2 ), ZerosMatrix = zeros( 2 ), Identity = eye( 2 ),
  Identity23 = eye( 2, 3 ), Identity32 = eye( 3, 2 )
```

输出结果：

```
OnesMatrix =
    1    1
    1    1
ZerosMatrix =
    0    0
    0    0
Identity =
    1    0
    0    1
Identity23 =
    1    0    0
    0    1    0
Identity32 =
    1    0
    0    1
    0    0
```

在命令行窗口中输入：

```
Random = rand( 2, 3 ), Array = Random( :, 2 ), Diagelement = diag( Random ),
  Diagmatrix = diag( diag( Random ) ), Dmatrix_array = diag( Array ),
  UpperTriangular = triu( Random ), LowerTriangular = tril( Random )
```

输出结果：

```
Random =
   0.2785   0.9575   0.1576
   0.5469   0.9649   0.9706
Array =
   0.9575
   0.9649
Diagelement =
   0.2785
   0.9649
Diagmatrix =
   0.2785        0
        0   0.9649
Dmatrix_array =
   0.9575        0
        0   0.9649
UpperTriangular =
   0.2785   0.9575   0.1576
        0   0.9649   0.9706
LowerTriangular =
   0.2785        0        0
   0.5469   0.9649        0
```

3．向量、标量和空矩阵

通常情况下，矩阵包含 m 行和 n 列，即 $m \times n$。当 m 和 n 取一些特殊值时，得到的矩阵具有一些特殊的性质。

（1）向量。

当 m=1 或 n=1，即 $1 \times n$ 或 $m \times 1$ 时，建立的矩阵称为向量。例如输入：

```
a = [1 2 3 4 5 6]
b = [1;2;3;4;5;6]
whos      % 调用 whos 函数查看变量 a、b 的相关信息
```

得到结果：

```
a =    1    2    3    4    5    6
b =
    1
    2
    3
    4
    5
    6
Name         Size             Bytes    Class    Attributes
  a      1x6                48       double
  b      6x1                48       double
```

（2）标量。

当 m=n=1 时，建立的矩阵称为标量。任意以 1×1 的矩阵形式表示的单个实数、复数都是标量。

例 2-33，在 MATLAB 中，标量有两种表示方法。

例如，依次输入：

```
x = 10+2i             % 将复数 10+2i 赋值给变量 x
shape = size(x)       % 查询变量 x 的形状信息
y = [10+2i]           % 将复数 10+2i 构成的矩阵赋值给变量 y
shape = size(y)       % 查询变量 y 的形状信息
x==y                  % 判断变量 x 和 y 是否相等，“1”表示相等，“0”表示不相等
```

得到结果：

```
x = 10.000000000000000 + 2.000000000000000i
shape =    1    1
y = 10.000000000000000 + 2.000000000000000i
shape =    1    1
ans =    1
```

通过上述示例我们可以知道：单个实数或者复数在 MATLAB 中都是以矩阵的形式存储的；在 MATLAB 中，单个数据或由单个数据构造的矩阵都是标量。

（3）空矩阵。

当 m=n=0，或者 m=0，或者 n=0，即 0×0、$0 \times n$、$m \times 0$ 时，创建的矩阵称为空矩阵。空矩阵可以通过赋值语句建立。例如输入：

```
x = []                % 建立一个空矩阵
whos x                % 调用 whos 函数查看变量 x 的相关信息
```

得到结果：

```
x =    []
  Name        Size              Bytes    Class    Attributes
  x           0x0           0        double
```

而再建立一个 0 矩阵，可以输入：

```
z = [0 0 0;0 0 0]     % 建立一个 2 行 3 列的 0 矩阵
whos z                % 调用 whos 函数查看变量 z 的相关信息
```

得到结果：

```
z =
    0    0    0
    0    0    0
  Name     Size          Bytes  Class    Attributes
  z       2x3              48  double
```

空矩阵和 0 矩阵的本质区别在于：空矩阵内没有任何元素，因此不占用任何存储空间；而 0 矩阵表示该矩阵中的所有元素全部为 0，需要占用一定的存储空间。

2.2.3　矩阵大小及结构的改变

根据运算时的不同情况和需要，矩阵大小和结构的改变，主要的方式有旋转矩阵、改变矩阵维度、删除矩阵元素等。MATLAB 中提供的具体此类函数如表 2-13 所示。

表 2-13　矩阵旋转与改变维度的函数

函数名称	函数功能
fliplr(***A***)	矩阵每一行均进行逆序排列
flipud(***A***)	矩阵每一列均进行逆序排列
flipdim(***A***, dim)	生成一个在 dim 维矩阵 ***A*** 内的元素交换位置的多维矩阵
rot90(***A***)	生成一个由矩阵 ***A*** 逆时针旋转 90°而得到的新矩阵
rot90(***A***, *k*)	生成一个由矩阵 ***A*** 逆时针旋转 *k*×90°而得到的新矩阵
reshape(***A*** , *m* , *n*)	生成一个 *m*×*n*×⋯×*p* 维的矩阵，其元素以线性索引的顺序从矩阵 ***A*** 中取得 如果矩阵 ***A*** 中没有 *m*×*n*×⋯×*p* 个元素，将返回一个错误信息
repmat(***A***,[*m n*⋯*p*])	创建一个和矩阵 ***A*** 有相同元素的 ***m***×***n***×⋯×***p*** 块的多维矩阵
shiftdim(***A*** , *n*)	矩阵的列移动 *n* 步。*n* 为正数，矩阵向左移；*n* 为负数，矩阵向右移
squeeze(***A***)	返回没有空维的矩阵 ***A***
cat(dim,***A***, ***B***)	将矩阵 ***A*** 和 ***B*** 组合成一个 dim 维的多维矩阵
permute(***A***, order)	根据向量 order 来改变矩阵 ***A*** 中的维数顺序
ipermute(***A***, order)	进行命令 permute 的逆变换
sort(***A***)	对一维或二维矩阵进行升序排序，并返回排序后的矩阵； 当 ***A*** 为二维矩阵时，对矩阵的每一列分别进行排序
sort(***A***, dim)	对矩阵按指定的方向进行升序排序，并返回排序后的矩阵。当 dim=1 时，对矩阵的每一列排序；dim=2 时，对矩阵的每一行排序
sort(***A***, dim, mode)	mode 为'ascend'时，进行升序排序；mode 为'descend'时，进行降序排序
[***B***, IX] = sort(***A***,⋯)	IX 为排序后备元素在原矩阵中的行位置或列位置的索引

例 2-34，矩阵的旋转与维度的改变。

在命令行窗口中输入：

```
Randoma = randn( 1,4 ), Randomb = randn( 2 ), Randoma = reshape( Randoma, 2,
    2 ), Randoma = fliplr( Randoma ), Randoma = rot90( Randoma ),
    Randomc = cat( 2, Randoma, Randomb ),
```

输出结果：

```
Randoma =                    %生成随机 1×4 矩阵
   0.6715   -1.2075   0.7172   1.6302
Randomb =                    %生成随机 2×2 矩阵
   0.4889    0.7269
   1.0347   -0.3034
Randoma =                    %将 1×4 矩阵变维为 2×2 矩阵
   0.6715    0.7172
  -1.2075    1.6302
Randoma =                    %将 2×2 矩阵每行逆序排列
   0.7172    0.6715
   1.6302   -1.2075
Randoma =                    %将 2×2 矩阵逆时针旋转 90°
   0.6715   -1.2075
   0.7172    1.6302
Randomc =                    %将两个 2×2 矩阵组合为一个 2×4 矩阵
   0.6715   -1.2075          0.4889       0.7269
 0.7172      1.6302          1.0347   -0.3034
```

2.2.4 矩阵下标引用

在 MATLAB 中，普通二维数组元素的数字索引分为双下标索引和单下标索引。双下标索引是通过一个二元数组对来对应元素在矩阵中的行列位置，例如 **A**(2,3)表示矩阵 **A** 中第 2 行第 3 列的元素。单下标索引的方式是采用列元素优先的原则，对 *m* 行 *n* 列的矩阵按列排序进行重组，成为一维数组，再取新的一维数组中的元素位置对应的值作为元素在原矩阵中的单下标，例如对于 4×4 的矩阵，**A**(7)表示矩阵 **A** 中第 3 行第 2 列的元素，而 **A**(13)表示矩阵 **A** 中第 1 行第 4 列的元素。

1. 矩阵下标访问单个矩阵元素

常用的矩阵索引表达式如表 2-14 所示。

表 2-14 矩阵的索引表达式

索引表达式	函数功能
A (l)	将二维矩阵 **A** 重组为一维数组，返回数组中第一个元素
A (:, *j*)	返回二维矩阵 **A** 中第 *j* 列列向量
A(*i* , :)	返回二维矩阵 **A** 中第 *i* 行行向量
A (: , *j* : *k*)	返回由二维矩阵 **A** 中的第 *j* 列到第 *k* 列列向量组成的子矩阵
A (*i* : *k* , :)	返回由二维矩阵 **A** 中的第 *i* 行到第 *k* 行行向量组成的子矩阵
A (*i* : *k* , *j* : *l*)	返回由二维矩阵 **A** 中的第 *i* 行到第 *k* 行行向量 和第 *j* 列到第 l 列列向量的交集组成的子矩阵

（续表）

索引表达式	函数功能
A(:)	将矩阵 **A** 中的每列合并成一个长的列向量
A($j:k$)	返回一个行向量，其元素为 **A**(:)中的第 j 个元素到第 k 个元素
A([j_1 j_2 …])	返回一个行向量，其元素为 **A**(:)中的第 j_1、j_2 元素
A(:,[j_1 j_2…])	返回矩阵 **A** 的第 j_1 列、第 j_2 列等的列向量
A([i_1 i_2 …], :)	返回矩阵 **A** 的第 i_1 行、第 i_2 行等的行向量
A([i_1 i_2…],[j_1 j_2 …])	返回矩阵第 i_1 行、第 i_2 行等和第 j_1 列、第 j_2 列等的元素

例 2-35，矩阵下标的引用示例。

在命令行窗口中输入：

```
Matrix = magic( 6 ) , Submatrix = Matrix( 2:3, 3:6 ), Array =
        Matrix( [7:10 26:31] )
```

输出结果：

```
Matrix =
   35    1    6   26   19   24
    3   32    7   21   23   25
   31    9    2   22   27   20
    8   28   33   17   10   15
   30    5   34   12   14   16
    4   36   29   13   18   11
Submatrix =
    7   21   23   25
    2   22   27   20
Array =
    1   32    9   28   23   27   10   14   18   24
```

2. 线性引用矩阵元素

矩阵中某一元素的单下标索引值和双下标索引值之间，可以通过 MATLAB 内部函数进行转换，其句法形式为：

```
IND = sub2ind(siz, i, j)
```

其功能为将双下标索引值转换为单下标索引值，其中 siz 是一个包含两个元素的数组，代表了转换矩阵的行列数，一般可以直接用 size(A)表示，i 与 j 分别是双下表索引中的行、列值，IND 是转换后的单下标索引值。

```
[I J] = ind2sub(siz, ind)
```

其功能为将单下标索引值转换为双下标索引值，各变量意义同上。

例 2-36，矩阵元素单双下标索引值转换示例。

在命令行窗口中输入：

```
Matrix = magic( 3 ); IND = sub2ind( size( Matrix ), 2,3),
   [I J]= ind2sub( size( Matrix ), 7)
```

输出结果：

```
IND =    8
I =    1
J =    3
```

3．访问多个矩阵元素

设 ***A***=magic(4)，如果需要计算第 4 列元素的和，按照前面介绍的方法可以用以下表达式来实现：

$$\boldsymbol{A}(1,4)+\boldsymbol{A}(2,4)+\boldsymbol{A}(3,4)+\boldsymbol{A}(4,4)$$

在下标表达式里，可以用冒号来表示矩阵的多个元素。例如，***A*** (1:*k*,*j*)表示矩阵第 *j* 列的前 *k* 个元素。利用冒号，第 4 列元素的和可以用更为简洁的式子，代码设置如下：

```
sum(A(1:4, 4))
```

还有更简洁的方法，因为冒号本身可以表示一列或者一行的所有元素，所以上式还可以写为：

```
sum(A(:,4))
```

在 MATLAB 中提供了一个关键字 end，用于表示该维中的最后一个元素，所以上式还可以改写成：

```
sum(A(:,end))
```

实际上还可以用冒号来表示非相邻的多个元素，例如下面的代码：

```
A=1:10
B=A(1:3:10)
```

由上述语句得到如下输出结果：

```
A=1 2 3 4 5 6 7 8 9 10
B=1 4 7 10
```

2.2.5 矩阵信息的获取

矩阵的信息主要包括矩阵结构、矩阵大小、矩阵维度、矩阵的数据类型及内存占用等。

1．矩阵结构

矩阵的结构是指矩阵子元素的排列方式。MATLAB 提供了各种测试函数，如表 2-15 所示。

表 2-15 矩阵结构测试函数

函数名称	函数功能
isempty(***A***)	检测矩阵是否为空
isscalar(***A***)	检测矩阵是否是单元素的标量矩阵
isvector(***A***)	检测矩阵是否是只具有一行或一列元素的一维向量
issparse(***A***)	检测数组是否是稀疏矩阵

这类函数的返回值是逻辑类型的数据。返回值为“1”表示该矩阵是某一特定类型的矩阵；返回值为“0”表示该矩阵不是该特定类型的矩阵。

例 2-37，矩阵数据结构判断函数的使用方法示例。利用 zeros 函数生成一个 4×4、元素全为 0 的矩阵 ***A***，并判断矩阵 ***A*** 的数据结构。首先输入：

```
A = zeros(4,4)
```

得到：

```
A =
    0    0    0    0
    0    0    0    0
    0    0    0    0
    0    0    0    0
```

再利用表 2-15 中的各函数判断矩阵 ***A*** 的数据结构，依次输入的代码和结果分别如下：

```
isempty(A) % 判断矩阵 A 是否为空矩阵
ans =    0
isscalar(A) % 判断矩阵 A 是否为标量
ans =    0
isvector(A) % 判断矩阵 A 是否为向量
ans =    0
issparse(A) % 判断矩阵 A 是否为稀疏矩阵
ans =    0
```

2. 矩阵大小

矩阵的形状信息反映了矩阵的大小，通常又包括以下几个方面的内容：

- 矩阵的维数。
- 矩阵各维（例如最长维、用户指定的维）的长度。
- 矩阵元素的个数。

针对上述 3 个方面的信息，相应地，MATLAB 提供了 4 个函数，分别用于获取矩阵形状的相关信息，如表 2-16 所示。

表 2-16　矩阵形状信息的查询函数

函　　数	调用格式	描　　述
ndims	*n*=ndims(***X***)	获取矩阵的维数
size	[*m*,*n*]=size(***X***)	获取矩阵在各维上的长度
length	*n*=length(***X***)	获取矩阵最长维的长度
numel	*n*=numel(***X***)	获取矩阵元素的个数

例 2-38，矩阵形状信息查询函数的使用示例。下面程序利用 eye 函数建立一个 5×3 的矩阵，利用 ndims 函数获得矩阵 ***A*** 的维数信息，输入的程序和得到的输出结果依次如下：

```
A = eye(5,3)
A =
    1    0    0
    0    1    0
    0    0    1
    0    0    0
    0    0    0
```

下面利用 ndims 函数获得矩阵 ***A*** 的维数信息。

```
ndims(A)
ans =    2
```

利用 length 函数获取矩阵 ***A*** 最长维的长度。

```
length(A)
ans =    5
```

利用 size 函数获取矩阵 ***A*** 各维的长度。

```
[m,n] = size(A)
m =    5
n =    3
d = size(A)
d =    5    3
e1 = size(A,1)
e1 =    5
e2 = size(A,2)
e2 =    3
```

由上述 size 函数的应用可知：①size 函数的返回值可以是分开显示的单个实数变量，也可以是一个行向量；②在 size 函数的输入参数中增加维度参数可以获取指定维度的长度，其中“1”表示行，“2”表示列。

使用 numel 函数（numel 是 number of elements 的简写）可以获取矩阵 ***A*** 中元素的个数。例如，输入：

```
f = numel(A)
```

得到的结果如下：

```
f =   15
```

例 2-39，数值与矩阵的算术运算示例。

在命令行窗口中输入：

```
A = []; B = 1:4; C = [1:4; 5:8];
S1= size(A), S2= size(B), S3= length(B), S4= size(C), S5= length(C), S6= numel(C)
```

输出结果：

```
S1 =    0    0
S2 =    1    4
S3 =    4
S4 =    2    4
S5 =    4
S6 =    8
```

3. 矩阵维度

对于空矩阵、标量矩阵、一维矩阵和二维矩阵，MATLAB 都将其作为普通二维数组对待。特别需要注意的是，用[]产生的空矩阵是作为二维矩阵的，但是在高维矩阵中也有空矩阵的概念，此时空矩阵则具有多个维度。

MATLAB 中提供了 ndims 函数计算矩阵维度。

例 2-40，矩阵维度示例。

在命令行窗口中输入：

```
A = []; B = 5; C = 1:3; D = magic(2); E(:,:,2) = [1 2; 3 4];
Ndims = [ndims(A) ndims(B) ndims(C) ndims(D) ndims(E)]
```

输出结果：

```
Ndims =    2    2    2    2    3
```

4. 矩阵的数据类型

矩阵作为 MATLAB 的内部数据存储和运算结构，其元素可以是各种各样的数据类型（具

体见 2.1 节），对应不同数据类型的元素，可以是数值、字符串、元胞、结构体等。MATLAB 中提供了一系列关于数据类型的测试函数，如表 2-17 所示。

表 2-17　矩阵数据类型的测试函数

函数名称	函数功能
isnumeric	检测矩阵元素是否为数值型变量
isreal	检测矩阵元素是否为实数数值型变量
isfloat	检测矩阵元素是否为浮点数值型变量
isinteger	检测矩阵元素是否为整数型变量
islogical	检测矩阵元素是否为逻辑型变量
ischar	检测矩阵元素是否为字符型变量
isstruct	检测矩阵元素是否为结构体型变量
iscell	检测矩阵元素是否为元胞型变量
iscellstr	检测矩阵元素是否为结构体的元胞型变量

这类函数的返回值也是逻辑类型的数据。返回值为“1”表示是某一特定的数据类型；返回值为“0”表示不是该特定的数据类型。

例 2-41，矩阵元素的数据类型的判断示例。

程序如下所示：

```
A = [2 3;10 7]
isnumeric(A)
isfloat(A)
islogical(A)
```

得到的结果为：

```
A =    2    3
         10    7
ans =    1
ans =    1
ans =    0
```

建立一个字符串矩阵 ***B*** 并进行判断，程序如下所示：

```
B = ['MATLAB';'course']
isstruct(B)
ischar(B)
```

得到的结果为：

```
B = MATLAB
course
ans =    0
ans =    1
```

例 2-42，矩阵的数据类型示例。

在命令行窗口中输入：

```
Mat = magic(2);
TrueorFalse = [isnumeric(Mat) isinteger(Mat) isreal(Mat) isfloat(Mat)]
```

输出结果：

```
TrueorFalse =
    1    0    1    1
```

例 2-43，将矩阵 ***A*** 中的实数和复数分开为一个具有实数和复数的矩阵示例。

在命令行窗口中输入：

```
clear all
%定义一个具有实数和复数的矩阵
A=[2 6.5 3i 3.5 6 4+2i];
%定义存储实数和复数的矩阵目前为空矩阵
real_array=[];
complex_array=[];
for i=1:length(A),
    %判断矩阵元素是否为实数
    if isreal(A(i))==1,
        real_array=[real_array A(i)];
        else
        complex_array=[complex_array A(i)];
        end;
end;
%输出实数元素
real_array
%输出复数元素
complex_array
```

输出结果：

```
real_array =
   2.0000    6.5000    3.5000    6.0000
complex_array =
       0 + 3.0000i   4.0000 + 2.0000i
```

5. 矩阵占用的内存

了解矩阵的内存占用情况，对于优化 MATLAB 代码性能是十分重要的。使用者可以通过 whos 命令查看当前工作区中指定变量的所有信息，包括变量名、矩阵大小、内存占用和数据类型等。

例 2-44，查看矩阵占用的内存示例。

在命令行窗口中输入：

```
Matrix = rand(2)
```

输出结果：

```
Matrix =
   0.2551    0.6991
   0.5060    0.8909
```

在命令行窗口中输入：

```
whos Matrix
```

输出结果：

```
  Name        Size            Bytes  Class     Attributes
  Matrix      2x2                32  double
```

2.2.6　矩阵的保存和加载

设有矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$，现用户希望将元素 1 修改成-1，并将 5、6、8、9 这几个矩阵元素以 0 来替代，之后在原始矩阵的最后添加一行，使得新的矩阵 ***A*** 满足某方面的使用要求。因此，用户首先要通过一定的途径让 MATLAB 找到元素“1”及元素区域“5、6、8、9”，接着将这些矩阵元素的值修改，最后增加一行。上述几个方面构成了矩阵元素的基本操作。

本节将介绍矩阵在 MATLAB 中的存储方式，接着介绍矩阵元素的寻址方法，最后介绍矩阵元素的赋值及扩展。

本节的重点在于让用户理解 MATLAB 存储矩阵的方法，进而掌握矩阵元素的寻址方法。

1．矩阵在 MATLAB 中的存储方法

设有矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 0 & -1\\ 2 & 4 & 9\\ -5 & 3 & 0\end{bmatrix}$，用户可以通过在 MATLAB 的命令行窗口内直接输入矩阵元素来创建此矩阵：

```
A = [1 0 -1;2 4 9;-5 3 0]
```

命令行窗口输出如下信息：

```
A =
     1     0    -1
     2     4     9
    -5     3     0
```

事实上，MATLAB 并不是按照其命令行输出的格式将矩阵 ***A*** 存储在内存空间中的。我们可以把内存空间想象成一列网格



而 MATLAB 就是将矩阵元素按列优先排列的原则依次放置在相应的格子内。因此，可以看成是一个长列向量，即



1	**2**	**-5**	**0**	**4**	**3**	**-1**	**9**	**0**

例如，矩阵第 2 行第 2 列的元素“4”，实际上位于存储空间的第 5 个格子的位置上。由此，MATLAB 采用了两种元素寻址方式：①矩阵下标寻址；②线性寻址。

2．矩阵元素的寻址

（1）矩阵下标寻址方法。

在 MATLAB 中，使用 ***A***(*i*,*j*)来表示一个矩阵 A 从左上角数起的第 *i* 行、第 *j* 列上的元素，这就是矩阵下标寻址方法。这种方法和线性代数中矩阵元素的引用方法一致，通俗易懂。以

下分别介绍利用矩阵下标寻址方法访问矩阵中的单个元素和元素区域。

1）单元素的寻访。

当使用双下标访问一个二维矩阵中的某个元素时，必须同时指定该元素所在的行号和列号，访问格式为：

```
A(numRow,numColumn)
```

其中，numRow 和 numColumn 分别代表行号和列号。

例 2-45，单个矩阵元素的访问示例。利用 rand 函数创建一个 4×3 的 0～1 均匀分布的随机数矩阵 ***A*** 并访问其中的元素。

输入的程序如下所示：

```
A = rand(4,3)
x = A(2,2)
y = A(4,3)
```

得到的结果为：

```
A =
    0.957166948242946   0.421761282626275   0.655740699156587
    0.485375648722841   0.915735525189067   0.035711678574190
    0.800280468888800   0.792207329559554   0.849129305868777
    0.141886338627215   0.959492426392903   0.933993247757551
x =   0.915735525189067
y =   0.933993247757551
```

2）矩阵元素区域的寻访。

访问矩阵的多个元素，可以是某一行、某一列或者其中的部分元素，也可以是矩阵中的某一块区域。在 MATLAB 中，元素区域的访问需要用到冒号“:”来表示矩阵中的多个元素，具体的访问格式如下：

- A(1:m,n)：表示访问第 ***n*** 列的第一个元素至第 ***m*** 个元素。
- A(m,:) ：表示访问第 ***m*** 行的所有元素。
- A(i:j,m:n) ：表示访问从第 ***i*** 行至第 ***j*** 行、从第 ***m*** 列至第 ***n*** 列的矩阵区域。
- A(i:inc1:j,m:inc2:n) ：表示访问从第 ***i*** 行至第 ***j*** 行，行间隔为 inc1 行；从第 ***m*** 列至第 ***n*** 列，列间隔为 inc2 的非相邻的多个矩阵元素。

例 2-46，矩阵元素区域的访问。利用 randn 函数创建一个 10×8 的 0～1 正态分布随机矩阵 ***X***，并进行访问。

输入的程序如下所示：

```
X = randn(10,8)
A = X(2,:)
B = X(3:8,2:6)
```

得到的结果为：

```
X =
     0.2939    1.3703    1.0933   -0.2256   -1.0616   -0.1774    0.2157   -0.4390
    -0.7873   -1.7115    1.1093    1.1174    2.3505   -0.1961   -1.1658   -1.7947
     0.8884   -0.1022   -0.8637   -1.0891   -0.6156    1.4193   -1.1480    0.8404
    -1.1471   -0.2414    0.0774    0.0326    0.7481    0.2916    0.1049   -0.8880
    -1.0689    0.3192   -1.2141    0.5525   -0.1924    0.1978    0.7223    0.1001
```

```
   -0.8095   0.3129  -1.1135   1.1006   0.8886   1.5877   2.5855  -0.5445
   -2.9443  -0.8649  -0.0068   1.5442  -0.7648  -0.8045  -0.6669   0.3035
    1.4384  -0.0301   1.5326   0.0859  -1.4023   0.6966   0.1873  -0.6003
    0.3252  -0.1649  -0.7697  -1.4916  -1.4224   0.8351  -0.0825   0.4900
   -0.7549   0.6277   0.3714  -0.7423   0.4882  -0.2437  -1.9330   0.7394
A = -0.7873  -1.7115   1.1093   1.1174   2.3505  -0.1961  -1.1658  -1.7947
B =
   -0.1022  -0.8637  -1.0891  -0.6156   1.4193
   -0.2414   0.0774   0.0326   0.7481   0.2916
    0.3192  -1.2141   0.5525  -0.1924   0.1978
    0.3129  -1.1135   1.1006   0.8886   1.5877
   -0.8649  -0.0068   1.5442  -0.7648  -0.8045
   -0.0301   1.5326   0.0859  -1.4023   0.6966
```

（2）线性寻址。

线性寻址的原理来自于 MATLAB 将矩阵元素存储在内存空间的存储方法。与矩阵下标寻址相比，只需要单一下标即可实现矩阵中任意位置元素的寻访。线性寻址的下标是通过矩阵的双下标换算得到的。

一般，设 $\boldsymbol{A}$ 是一个 $m\times n$ 的矩阵，位于第 i 行、第 j 列上的元素 $\boldsymbol{A}(i,j)$的单一下标为 $\boldsymbol{A}((j\text{-}1)\cdot m+i)$。

例 2-47，线性寻址示例。建立 3 阶希尔伯特矩阵 $\boldsymbol{A}$，并进行线性寻址访问，程序如下：

```
A = hilb(3)
A(2,3) % 采用矩阵下标寻址的方法访问第 2 行第 3 列的元素
A(8)
```

得到的结果如下：

```
A =
   1.0000   0.5000   0.3333
   0.5000   0.3333   0.2500
0.3333   0.2500   0.2000
ans =   0.2500
ans =   0.2500
```

（3）矩阵元素的赋值。

MATLAB 使用赋值语句对矩阵元素进行赋值，基本语法如下：

- A(i,j) = value：等号左侧为矩阵中的某个元素；等号右侧为值。
- A = []：删除矩阵中的所有元素。

例 2-48，矩阵元素的赋值示例。利用 magic 函数建立一个 4 阶的魔幻矩阵 $\boldsymbol{M}$，并进行复制操作。

输入的命令如下所示：

```
M = magic(4)
M(2,1)=-2
M(3:4,3:4)=0
M = []
whos M % 调用 whos 函数查看矩阵 M 的详细信息
```

得到的结果为：

```
M =
   16    2    3   13
    5   11   10    8
```

```
     9     7     6    12
     4    14    15     1
M =
    16     2     3    13
    -2    11    10     8
     9     7     6    12
     4    14    15     1
M =
    16     2     3    13
    -2    11    10     8
     9     7     0     0
     4    14     0     0
M =     []
  Name        Size          Bytes   Class    Attributes
  M           0x0           0       double
```

（4）矩阵元素的扩展与删除。

增加或删除矩阵元素最常用的办法是使用赋值语句。

例 2-49，对于矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 1\\ 2 & 2\end{bmatrix}$，如果现在要增加一行，则应在命令行窗口内输入：

```
A = [1 1;2 2]
A(3,:) = 3                    % 整行赋值
A(4,1) = 4;A(4,2) = 5         % 单个矩阵元素赋值的办法增加新元素
A(2,:) = []                   % 使用空矩阵[]删除矩阵中的整行或整列
```

得到的结果为：

```
A =
     1     1
     2     2
A =
     1     1
     2     2
     3     3
A =
     1     1
     2     2
     3     3
     4     5
A =
     1     1
     3     3
     4     5
```

此外，MATLAB 提供了多个函数进行矩阵合并操作，从而实现将多个矩阵合并成一个矩阵。这些函数如表 2-18 所示。

表 2-18　矩阵合并函数

函　　数	基本调用格式	描　　述
cat	cat(DIM,**A**,**B**)	在 DIM 指定的维度上合并矩阵 **A** 和 **B**。DIM=1 表示按行（竖直方向）合并；DIM=2 表示按列（水平方向）合并
horzcat	horzcat(**A**,**B**)	在水平方向上合并矩阵 **A** 和 **B**
vertcat	vertcat(**A**,**B**)	在竖直方向上合并矩阵 **A** 和 **B**

（续表）

函　　数	基本调用格式	描　　述
repmat	***B*** = repmat(***A***,*M*,*N*)	通过复制 $M\times N$ 个矩阵 ***A*** 来构造新的矩阵 ***B***
blkdiag	***Y*** = blkdiag(***A***,***B***,...)	用已知的 ***A***、***B*** 等多个矩阵构造块对角化矩阵 ***Y***，其中 $\boldsymbol{Y}=\begin{bmatrix} \boldsymbol{A} & 0 & \cdots & 0 \\ 0 & \boldsymbol{B} & \cdots & 0 \\ 0 & 0 & \cdots & \boldsymbol{N} \end{bmatrix}$

例 2-50，矩阵合并函数示例。

（1）设有矩阵 $\boldsymbol{A}=\begin{bmatrix} 2 & 0 & -1 \\ 1 & 3 & 2 \end{bmatrix}$、矩阵 $\boldsymbol{B}=\begin{bmatrix} 1 & 7 & -1 \\ 4 & 2 & 3 \\ 2 & 0 & 1 \end{bmatrix}$、矩阵 $\boldsymbol{C}=\begin{bmatrix} 1 & 0 & 1 & 0 \\ -1 & 2 & 0 & 1 \end{bmatrix}$，利用 cat 函数分别对矩阵 ***A***、***B*** 及矩阵 ***A***、***C*** 进行合并操作，函数中的输入参数 DIM 分别取 DIM=1 及 DIM=2。

首先输入：

```
A = [2 0 -1;1 3 2];
B = [1 7 -1;4 2 3;2 0 1];
C = [1 0 1 0;-1 2 0 1];
```

得到矩阵 ***A***、***B***、***C***。然后使用 cat 函数对矩阵 ***A***、***B*** 进行合并操作。

```
MAB1 = cat(1,A,B) % 将矩阵 A、B 按行合并
```

得到结果：

```
MAB1 =
     2     0    -1
     1     3     2
     1     7    -1
     4     2     3
     2     0     1
```

如果合并错误，如：

```
MAB2 = cat(2,A,B) % 将矩阵 A、B 按列合并
MAC1 = cat(1,A,C) % 将矩阵 A、C 按行合并
```

得到结果：

```
Error using cat
CAT arguments dimensions are not consistent.
```

使用 cat 函数对矩阵 ***A***、***C*** 进行合并操作，输入如下：

```
MAC2 = cat(2,A,C) % 将矩阵 A、C 按列合并
```

得到结果：

```
MAC2 =
     2     0    -1     1     0     1     0
     1     3     2    -1     2     0     1
```

为了探究矩阵 ***A***、***B*** 按列合并及矩阵 ***A***、***C*** 按列合并发生错误的原因，将合并后的矩阵列出来：

$$[A \quad B]=\begin{bmatrix} 2 & 0 & -1 & 1 & 7 & -1 \\ 1 & 3 & 2 & 4 & 2 & 3 \\ & & & 2 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} A \\ C \end{bmatrix}=\begin{bmatrix} 2 & 0 & -1 & \\ 1 & 3 & 2 & \\ & & & \\ 1 & 0 & 1 & 0 \\ -1 & 2 & 0 & 1 \end{bmatrix}$$

可以看出，由于两个矩阵在某个维度上的长度不一致，导致新的矩阵出现残缺。

因此，cat 及其相关的函数在将两个矩阵按某个维度进行合并操作时，原始的两个矩阵要在某一个维度上具有相同的长度，否则 MATLAB 在进行计算时就会发生错误。

（2）使用 blkdiag 函数构造块对角化矩阵。例如输入：

```
A = [1 1;2 2];B=[3 3;4 4];C=[5 5;6 6];
Y = blkdiag(A,B,C)
```

得到结果：

```
Y =
    1    1    0    0    0    0
    2    2    0    0    0    0
    0    0    3    3    0    0
    0    0    4    4    0    0
    0    0    0    0    5    5
    0    0    0    0    6    6
```

2.3 运算符

MATLAB 中的运算符分为算术运算符、关系运算符和逻辑运算符。这三种运算符可以分别使用，也可以在同一运算式中出现。当在同一运算式中同时出现两种或两种以上运算符时，运算的优先级排列如下：算术运算符优先级最高，其次是关系运算符，最低级别是逻辑运算符。

2.3.1 算术运算符

MATLAB 中的算术运算符有加、减、乘、除、点乘、点除等，其运算法则如表 2-19 所示。

表 2-19 MATLAB 中的算术运算符

算术运算符	运算法则	算术运算符	运算法则
A*+*B	***A*** 与 ***B*** 相加 （***A***、***B*** 为数值或矩阵）	***A* - *B***	***A*** 与 ***B*** 相减 （***A***、***B*** 为数值或矩阵）
A* * *B	***A*** 与 ***B*** 相乘 （***A***、***B*** 为数值或矩阵）	***A*.* *B***	***A*** 与 ***B*** 相应元素相乘 （***A***、***B*** 为相同维度的矩阵）
A* / *B	***A*** 与 ***B*** 相除 （***A***、***B*** 为数值或矩阵）	***A*./ *B***	***A*** 与 ***B*** 相应元素相除 （***A***、***B*** 为相同维度的矩阵）
A* ^ *B	***A*** 的 ***B*** 次幂 （***A***、***B*** 为数值或矩阵）	***A*.^ *B***	***A*** 的每个元素的 ***B*** 次幂 （***A*** 为矩阵，***B*** 为数值）

例 2-51，数值与矩阵的算术运算示例。

在命令行窗口中输入：

```
A = eye( 2 ), B = ones( 2 ), C = A * B, D = A .* B
```

输出结果：

```
A =
     1    0
     0    1
B =
     1    1
     1    1
C =                 %A 与 B 两个矩阵相乘
     1    1
     1    1
D =                 %A 与 B 两个矩阵的每个元素分别相乘
     1    0
     0    1
```

MATLAB 平台上还提供了大量的运算函数，其中的常用运算函数如表 2-20 所示。

表 2-20　MATLAB 中的常用运算函数

函　数	运算法则
exp(*x*)	求以 e 为底数的 *x* 次幂
log(*x*)	以 e 为底数对 *x* 取对数
log10(*x*)	以 10 为底数对 *x* 取对数
sqrt(*x*)	*x* 的平方根
sin(*x*)	*x* 的正弦函数
cos(*x*)	*x* 的余弦函数
tan(*x*)	*x* 的正切函数
asin(*x*)	*x* 的反正弦函数
acos(*x*)	*x* 的反余弦函数
atan(*x*)	*x* 的反正切函数
mod(*a*,*b*)	*a* 与 *b* 相除取余数
min(*a*,*b*)	返回 *a* 与 *b* 中较小的数值
max(*a*,*b*)	返回 *a* 与 *b* 中较大的数值
mean(*x*)	找出 *x* 阵列的平均值
median(*x*)	找出 *x* 阵列的中位数
sum(*x*)	计算 *x* 阵列的总和值
prod(*x*)	计算 *x* 阵列的连乘值
cumsum(*x*)	计算 *x* 阵列的累积总和值
cumprod(*x*)	计算 *x* 阵列的累积连乘值
sign(*x*)	*x* <0 时返回值为-1，*x* =0 时返回值为 0，*x* >0 时返回值为 1
rem(*x*,*y*)	返回 *x*/*y* 的余数
diff(*x*)	***x*** 向量的差分
sort(x)	对 ***x*** 向量进行排序
fft(x)	***x*** 向量的离散傅里叶变换
rank(x)	***x*** 矩阵的秩

2.3.2 关系运算符

MATLAB 中的关系运算符有 6 个，其运算法则如表 2-21 所示。

表 2-21 MATLAB 中的关系运算符

关系运算符	关系说明	关系运算符	关系说明
<	小于	<=	小于等于
>	大于	>=	大于等于
==	等于	~=	不等于

注意

"="和"= ="的区别："= ="的运算法则是比较两个变量，当它们相等时返回 1，当它们不相等时返回 0；而"="则是被用来将运算的结果赋给一个变量。

关系运算符可以用来对两个数值、两个数组、两个矩阵或两个字符串等数据类型进行比较，同样也可以进行不同类型两个数据之间的比较。比较的方式根据所比较的两个数据类型的不同而不同。例如对矩阵和一个标量进行比较时，需将矩阵中的每个元素与标量进行比较。

关系运算符通过比较对应的元素，产生一个仅包含 1 和 0 的数值或矩阵。其元素代表的意义如下：

- 返回值为 1，比较结果是真。
- 返回值为 0，比较结果是假。

例 2-52，关系运算符的运用。

在命令行窗口中输入：

```
A=1:9, B=10-A
```

输出结果：

```
A =    1    2    3    4    5    6    7    8    9
B =       9    8    7    6    5    4    3    2    1
```

在命令行窗口中输入：

```
TrueorFalse = ( A>4 )
```

输出结果：

```
TrueorFalse =   0    0    0    0    1    1    1    1    1
```

在命令行窗口中输入：

```
TrueorFalse = ( A==B )
```

输出结果：

```
TrueorFalse =   0    0    0    0    1    0    0    0    0
```

例 2-53，关系运算符的运算。

在命令行窗口中输入：

```
C=5:-1:0;
```

```
C=C+(C==0)*eps
```

输出结果：

```
C =      5.0000    4.0000    3.0000    2.0000    1.0000    0.0000
```

○ 提示

上例中利用特殊的 MATLAB 数 eps 代替一个数组中的零元素，eps 近似为 2.2e-16。这种特殊的表达式在避免 0 作为分母时是很有用的。

2.3.3　逻辑运算符

逻辑运算符提供了一种组合或否定关系表达式。MATLAB 逻辑运算符如表 2-22 所示。

表 2-22　MATLAB 中的逻辑运算符

逻辑运算符	说　明
&	与
|	或
~	非

例 2-54，逻辑运算符的运用。

在命令行窗口中输入：

```
A=1:9
TrueorFalse = ~( A>4 )
```

输出结果：

```
TrueorFalse =    1    1    1    1    0    0    0    0    0
```

在命令行窗口中输入：

```
TrueorFalse =(A>2)&(A<6)
```

输出结果：

```
TrueorFalse =    0    0    1    1    1    0    0    0    0
```

与关系运算符一样，逻辑运算符也可以进行矩阵与数值之间的比较，比较的方式为将矩阵的每一个元素都与数值进行比较，比较结果为一个相同维数的矩阵，新生成矩阵的每一个元素都代表着原来矩阵中相同位置上的元素与该数值的逻辑运算结果。

使用逻辑运算符比较两个相同维数的矩阵时，是按元素来进行比较的，其比较结果是一个包含 1 和 0 的矩阵。元素 0 表示逻辑为假，元素 1 表示逻辑为真。

A* & *B 返回一个与 ***A*** 和 ***B*** 相同维数的矩阵。在这个矩阵中，***A*** 和 ***B*** 对应元素都为非零时，则对应项为 1；有一个为零时，则对应项为 0。

A* | *B 返回一个与 ***A*** 和 ***B*** 相同维数的矩阵。在这个矩阵中，***A*** 和 ***B*** 对应元素只要有一个为非零，则对应项为 1；两个矩阵均为零时，则对应项为 0。

~*A* 返回一个与 ***A*** 相同维数的矩阵。在这个矩阵中，***A*** 中对应项是零时，则对应项为 1；***A*** 中对应项是非零时，则对应项为 0。

除了上面的逻辑运算符，MATLAB 还提供了各种逻辑函数，基本的逻辑函数如表 2-23 所示。

表 2-23　MATLAB 中的重要逻辑函数

函　　数	运算法则
xor(*x*,*y*)	异或运算。*x* 与 *y* 不同时，返回 1；*x* 与 *y* 相同时，返回 0
any(*x*)	如果在一个向量 x 中，有任何元素是非零，返回 1；否则返回 0 如果矩阵 *x* 中的每一列有非零元素，返回 1；否则返回 0
all(*x*)	如果在一个向量 *x* 中，所有元素非零，返回 1；否则返回 0 矩阵 *x* 中的每一列所有元素非零，返回 1；否则返回 0

2.3.4　运算优先级

在一个表达式中，算术运算符优先级最高，其次是关系运算符，最后是逻辑运算符。需要时，可以通过加括号来改变运算顺序。MATLAB 中具体的运算优先级排列如表 2-24 所示。

表 2-24　运算优先级

优 先 级	运算法则
1	括号：　()
2	转置和乘幂：‘, ^, .^
3	一元加减运算和逻辑非：　+, -, ~
4	乘除，点乘，点除：　*, /, .*, ./
5	冒号运算：　:
6	关系运算：>, >=, <, <=, ==, ~=
7	逐个元素的逻辑与：&
8	逐个元素的逻辑或：　|
9	捷径逻辑与：&&
10	捷径逻辑或：　||

〇 提示

在表达式的书写中，建议采用括号分级的方式明确运算的先后顺序，避免优先级混乱而产生运算错误。

2.4　字符串处理函数

MATLAB 提供了大量的字符串函数，如表 2-25 所示。

表 2-25　MATLAB 中的字符串函数

字符串函数	函数功能
eval(string)	作为一个 MATLAB 命令求字符串的值
blanks(*n*)	返回一个 *n* 个零或空格的字符串
deblank	去掉字符串中后拖的空格
feval	求由字符串给定的函数值

（续表）

字符串函数	函数功能
findstr	从一个字符串内找出字符串
isletter	字母存在时返回真值
isspace	空格字符存在时返回真值
Isstr	输入一个字符串，返回真值
iasterr	返回上一个产生 MATLAB 错误的字符串
strcmp	字符串相同，返回真值
strrep	用一个字符串替换另一个字符串
strtok	在一个字符串里找出第一个标记

2.4.1　字符串的构造

字符串或字符串数组的构造可以通过直接给变量赋值来实现，具体表达式中字符串的内容需要写在单引号内。如果字符串的内容包含单引号，那么以两个重复的单引号来表示。

构造多行字符串时，若字符串内容写在[]内，那么多行字符串的长度必须相同；若字符串内容写在{}内，则这多行字符串的长度可以不同。

例 2-55，直接赋值构造字符串示例。

在命令行窗口中输入：

```
Str_a='How are you?', Str_b='I don''t know.', Str_c = strcat( Str_a, Str_b)
```

输出结果：

```
Str_a =How are you?
Str_b =I don't know.
Str_c =How are you?I don't know.
```

在命令行窗口中输入：

```
Str_mat = ['July';'August';'September';]
```

输出结果：

```
错误使用 vertcat
串联的矩阵的维度不一致
```

在命令行窗口中输入：

```
Str_mat1 = ['U r a man.'; 'I'm a pen.'],
Str_mat2 = {'July';'August';'September';}
```

输出结果：

```
Str_mat1 =
U r a man.
I'm a pen.
Str_mat2 =
    'July'
    'August'
    'September'
```

MATLAB 中还提供了 strvcat 和 char 函数用于纵向连接多个字符串。strvcat 函数连接多

行字符串时，每行字符串的长度不要求相等，所有非最长字符串的右边会自动补偿空格，使得每行字符串的长度相同。char 函数与 strvcat 函数类似，不过当多行字符串中有空字符串时，strvcat 函数会自动进行忽略，而 char 函数会把空字符串也用空格补偿后再进行连接。

例 2-56，构造字符串示例。

在命令行窗口中输入：

```
A='top';  B=''; C='Bottom';
sABC=strvcat(A,B,C),cABC=char(A,B,C),size=[size(sABC);size(cABC)]
```

输出结果：

```
sABC =
top
Bottom
cABC =
top

Bottom
size =
    2    6
    3    6
```

2.4.2　字符串比较函数

两个字符串之间的关系可以通过关系运算符来比较，也可以使用 strcmp 函数来比较两个字符串是否相同。

例 2-57，比较字符串示例。

在命令行窗口中输入：

```
 A = (' Hello ' == ' Word ')
```

输出结果：

```
??? Error using ==> eq
Matrix dimensions must agree.
```

在命令行窗口中输入：

```
A = (' Hello ' == ' World '), B = (' Hello ' == ' Hello '), C = strcmp(' Hello
  ', ' World '),  D = strcmp(' Hello ', ' Hello ')
```

输出结果：

```
A =    1    0    0    0    1    0    1
B =    1    1    1    1    1    1    1
C =    0
D =    1
```

○ 提示

使用关系运算符进行比较时，会对字符串的每个字符进行比较，返回值是一个与字符串长度相同大小的数组，因此被比较的两个字符串的长度必须相同；而 strcmp 函数则根据两个字符串相同与否，返回值为数值 0 或 1。

2.4.3　字符串查找和替换函数

字符串的查找与搜索可以通过 findstr 函数来实现。

例 2-58，按下标值查找字符串示例。

在命令行窗口中输入：

```
String =' Peter Piper picked a peck of pickled peppers. ' ;
findstr(String,'  ')  %搜索字符串内的空格位置
```

输出结果：

```
ans =    6    12    19    21    26    29    37
```

在命令行窗口中输入：

```
findstr(String,' p ')          %搜索字母 p
```

输出结果：

```
ans =    9    13    22    30    38    40    41
```

在命令行窗口中输入：

```
findstr(String, ' cow ')        %搜索单词 cow
```

输出结果：

```
ans =    [  ]
```

在命令行窗口中输入：

```
findstr(String,' pick ')       %搜索单词 pick
```

输出结果：

```
ans =   13    30
```

○ 提示

findstr 函数对字母的大小写是敏感的。另外，findstr 函数对字符串矩阵不起作用，因此对字符串矩阵的搜索只能通过循环索引矩阵内的元素进行搜索。

字符串的替换可以通过对字符串数组中相应的元素直接赋值来实现，也可以使用 strrep 函数来实现。

例 2-59，替换字符串示例。

在命令行窗口中输入：

```
String =' Peter Piper picked a peck of pickled peppers. ' ;
String(1:12) =' Helen Smith'
```

输出结果：

```
String =Helen Smith picked a peck of pickled peppers.
```

在命令行窗口中输入：

```
String = strrep( String, ' Helen Smith', ' Sabrina Crame')
```

输出结果：

```
String =Sabrina Crame picked a peck of pickled peppers.
```

> **提示**
>
> 直接赋值方法并不能使两个不同长度的字符串相互替换，而 strrep 函数可以替换两个任意长度的字符串。与 findstr 函数类似，strrep 函数也对字符串矩阵不起作用。

2.4.4 字符串——数值转换

MATLAB 还提供了大量字符串与各种数据类型之间的转换函数，如表 2-26 所示。

表 2-26 字符串类型与数字类型转换函数

函数名称	函数功能
abs	字符转换成 ASCII
dec2hex	十进制数转换成十六进制字符串
fprintf	把格式化的文本写到文件中或显示屏上
hex2dec	十六进制字符串转换成十进制数
hex2num	十六进制字符串转换成 IEEE 浮点数
int2str	整数转换成字符串
lower	字符串转换成小写
num2str	数字转换成字符串
setstr	ASCII 转换成字符串
sprintf	用格式式控制数字转换成字符串
sscanf	用格式式控制字符串转换成数字
str2mat	字符串转换成一个文本矩阵
str2num	字符串转换成数字
upper	字符串转换成大写

例 2-60，将数值嵌入字符串示例。

在命令行窗口中输入：

```
rad=2.5;  area=pi*rad ^2;
string =[' A circle of radius '  num2str(rad)  ' has an area of '
        num2str(area)  ' . ' ] ;
disp(string)
```

输出结果：

```
A circle of radius 2.5 has an area of 19.63.
```

2.5 本章小结

本章介绍了 MATLAB 的基础知识及一些基本编程的句法形式和函数的使用方法，对 MATLAB 中的数据类型、矩阵的基本操作、运算符和字符串操作函数分别进行了举例说明，其中矩阵、函数句柄和结构体等内容后续还有更详细的介绍。本章的重点是数值、字符串等的各种操作及相互之间的转换，读者还应熟练掌握各个运算符的运算法则及优先级。

第 3 章

数组与矩阵

知识要点

MATLAB 中的所有数据都按照数组的形式进行存储和运算，数组的属性及数组之间的逻辑关系是编写程序时非常重要的两个方面。在 MATLAB 平台上，数组的定义是广义的，数组的元素可以是任意的数据类型，例如数值、字符串等。矩阵是特殊的数组，在很多工程领域，都会遇到矩阵分析和线性方程组的求解等问题。本章的主要内容是对矩阵和数组的深入讲解，以及关于矩阵和数组的复杂运算的应用。

学习要求	学习目标 / 知识点	了解	理解	应用	实践
	数组的创建			√	
	数组的常见运算		√		√
	矩阵的创建			√	√
	矩阵的常见运算		√		√
	稀疏矩阵	√			√



3.1 数组运算

数组运算是 MATLAB 计算的基础。由于 MATLAB 面向对象的特性，这种数值数组成为 MATLAB 最重要的一种内建数据类型，而数组运算就是定义这种数据结构的方法。本节将系统地列出具备数组运算能力的函数名称，为兼顾一般性，以二维数组的运算为例，读者可推广至多维数组和多维矩阵的运算。

下面将介绍在 MATLAB 中如何建立数组，以及数组的常用操作等，包括数组的算术运算、关系运算和逻辑运算。

3.1.1 数组的创建和操作

在 MATLAB 中一般使用方括号“[ ]”、逗号“,”、空格号和分号“;”来创建数组，数组中同一行的元素使用逗号或空格进行分隔，不同行之间用分号进行分隔。

例 3-1，创建空数组、行向量、列向量示例。

在命令行窗口中输入如下语句：

```
clear all
A=[]
B=[6 5 4 3 2 1]
C=[6,5,4,3,2,1]
D=[6;5;4;3;2;1]
E=B'  %转置
```

输出结果：

```
A =    []
B =    6    5    4    3    2    1
C =    6    5    4    3    2    1
D =
    6
    5
     4
    3
    2
    1
E =
    6
    5
    4
    3
    2
    1
```

例 3-2，访问数组示例。

在命令行窗口中输入如下语句：

```
clear all
A=[6 5 4 3 2 1]
a1=A(1)            %访问数组第一个元素；
```

```
a2=A(1:3)         %访问数组第 1、2、3 个元素
a3=A(3:end)       %访问数组第 3 个到最后一个元素
a4=A(end:-1:1)    %数组元素反序输出
a5=A([1 6])       %访问数组第 1 个及第 6 个元素
```

输出结果：

```
A =      6    5    4    3    2    1
a1 =    6
a2 =    6    5    4
a3 =    4    3    2    1
a4 =    1    2    3    4    5    6
a5 =    6    1
```

例 3-3，子数组的赋值（Assign）示例。

在命令行窗口中输入如下语句：

```
clear all
A=[6 5 4 3 2 1]
A(3) = 0
A([1 4])=[1 1]
```

输出结果：

```
A =    6    5    4    3    2    1
A =    6    5    0    3    2    1
A =    1    5    0    1    2    1
```

在 MATLAB 中还可以通过其他各种方式创建数组，具体如下所示。

1．通过冒号创建一维数组

在 MATLAB 中，通过冒号创建一维数组的代码：

```
X=A:step:B
```

其中，*A* 是创建一维数组的第一个变量，step 是每次递增或递减的数值，直到最后一个元素和 *B* 的差的绝对值小于等于 step 的绝对值为止。

例 3-4，通过冒号创建一维数组示例。

在命令行窗口中输入如下语句：

```
clear all
A=2:6
B=2.1:1.5:6
C=2.1:-1.5:-6
D=2.1:-1.5:6
```

输出结果：

```
A =    2    3    4    5    6
B =   2.1000   3.6000   5.1000
C =   2.1000   0.6000  -0.9000  -2.4000  -3.9000  -5.4000
D =  Empty matrix: 1-by-0
```

2．通过 logspace 函数创建一维数组

MATLAB 常用 logspace ()函数创建一维数组，该函数的调用方式如下：

- y= logspace(a,b)：该函数创建行向量 $\boldsymbol{y}$，第一个元素为 10^a，最后一个元素为 10^b，形成总数为 50 个元素的等比数列。

- y = logspace(a,b,n)：该函数创建行向量 ***y***，第一个元素为 10^a，最后一个元素为 10^b，形成总数为 ***n*** 个元素的等比数列。

例 3-5，通过 logspace 函数创建一维数组示例。

在命令行窗口中输入如下语句：

```
clear all
format short;
A=logspace(1,2,20)
B=logspace(1,2,10)
```

输出结果：

```
A = 10.0000   11.2884   12.7427   14.3845   16.2378   18.3298   20.6914
   23.3572   26.3665   29.7635   33.5982   37.9269   42.8133   48.3293
   54.5559  61.5848   69.5193   78.4760   88.5867  100.0000
B = 10.0000   12.9155   16.6810   21.5443   27.8256   35.9381   46.4159
   59.9484   77.4264  100.0000
```

3．通过 linspace 函数创建一维数组

MATLAB 常用 linspace()函数创建一维数组，该函数的调用方式如下：

- y= linspace (a,b)：该函数创建行向量 ***y***，第一个元素为 ***a***，最后一个元素为 ***b***，形成总数为 100 个元素的线性间隔向量。
- y = linspace (a,b,n)：该函数创建行向量 ***y***，第一个元素为 ***a***，最后一个元素为 ***b***，形成总数为 ***n*** 个元素的线性间隔向量。

例 3-6，通过 linspace 函数创建一维数组示例。

在命令行窗口中输入如下语句：

```
clear all
format short;
A = linspace(1,100)
B = linspace(1,36,12)
C= linspace(1,36,1)
```

输出结果：

```
A =
  Columns 1 through 25
  1     2     3     4     5     6     7     8     9    10    11    12    13    14    15
    16    17    18    19    20    21    22    23    24    25
  Columns 26 through 50
 26    27    28    29    30    31    32    33    34    35    36    37    38    39
    40    41    42    43    44    45    46    47    48    49    50
  Columns 51 through 75
 51    52    53    54    55    56    57    58    59    60    61    62    63    64
    65    66    67    68    69    70    71    72    73    74    75
  Columns 76 through 100
 76    77    78    79    80    81    82    83    84    85    86    87    88    89
    90    91    92    93    94    95    96    97    98    99   100
B =1.0000    4.1818    7.3636   10.5455   13.7273   16.9091   20.0909   23.2727
   26.4545   29.6364   32.8182   36.0000
C =    36
```

3.1.2　数组的常见运算

1．数组的算术运算

数组的运算是从数组的单个元素出发，针对每个元素进行的运算。在 MATLAB 中，一维数组的基本运算包括加、减、乘、左除、右除和乘方。

数组的加减运算：通过格式 *A*+*B* 或 *A*-*B* 可实现数组的加减运算。但是运算规则要求数组 *A* 和 *B* 的维数相同。

> **提示**
>
> 如果两个数组的维数不相同，则将给出错误的信息。

例 3-7，数组的加减运算示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 5 6 8 9 6]
B=[9 85 6 2 4 0]
C=[1 1 1 1 1]
D=A+B            %加法
E=A-B            %减法
F=A*2
G=A+3            %数组与常数的加法
H=A-C
```

输出结果：

```
A =    1    5    6    8    9    6
B =    9   85    6    2    4    0
C =    1    1    1    1    1
D =   10   90   12   10   13    6
E =   -8  -80    0    6    5    6
F =    2   10   12   16   18   12
G =    4    8    9   11   12    9
错误使用  -
矩阵维度必须一致。
```

数组的乘除运算：通过格式“.*”或“./”可实现数组的乘除运算。但是运算规则要求数组 *A* 和 *B* 的维数相同。

乘法：数组 *A* 和 *B* 的维数相同，运算为数组对应元素相乘，计算结果与 *A* 和 *B* 是相同维数的数组。

除法：数组 *A* 和 *B* 的维数相同，运算为数组对应元素相除，计算结果与 *A* 和 *B* 是相同维数的数组。

右除和左除的关系：*A*./*B*=*B*.*A*，其中 *A* 是被除数，*B* 是除数。

> **提示**
>
> 如果两个数组的维数不相同，则将给出错误的信息。

例 3-8，数组的乘法示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 5 6 8 9 6]
B=[9 5 6 2 4 0]
C=A.* B %数组的点乘
D=A * 3 %数组与常数的乘法
```

输出结果：

```
A =    1    5    6    8    9    6
B =    9    5    6    2    4    0
C =    9   25   36   16   36    0
D =    3   15   18   24   27   18
```

例 3-9，数组的除法示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 5 6 8 9 6]
B=[9 5 6 2 4 0]
C=A.\B %数组和数组的左除
D=A./B %数组和数组的右除
E=A./3 %数组与常数的除法
F=A/3
```

输出结果：

```
A =    1    5    6    8    9    6
B =    9    5    6    2    4    0
C =   9.0000   1.0000   1.0000   0.2500   0.4444        0
D =   0.1111   1.0000   1.0000   4.0000   2.2500      Inf
E =   0.3333   1.6667   2.0000   2.6667   3.0000   2.0000
F =   0.3333   1.6667   2.0000   2.6667   3.0000   2.0000
```

通过乘方格式“.^”实现数组的乘方运算。数组的乘方运算包括：数组间的乘方运算、数组与某个具体数值的乘方运算，以及常数与数组的乘方运算。

例 3-10，数组的乘方示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 5 6 8 9 6]
B=[9 5 6 2 4 0]
C=A.^B %数组的乘方
D=A.^3 %数组与某个具体数值的乘方
E=3.^A %常数与数组的乘方
```

输出结果：

```
A =    1    5    6    8    9    6
B =    9    5    6    2    4    0
C =    1       3125      46656         64       6561          1
D =    1   125   216   512   729   216
E =    3        243        729       6561      19683        729
```

通过函数 dot()可实现数组的点积运算，但是运算规则要求数组 *A* 和 *B* 的维数相同，其调用格式如下：

- C= dot(A,B)。

- C = dot(A,B,dim)。

例 3-11，数组的点积示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 5 6 8 9 6]
B=[9 5 6 2 4 0]
C=dot(A,B)  %数组的点积
D=sum(A.*B) %数组元素的乘积之和
```

输出结果：

```
A =    1    5    6    8    9    6
B =    9    5    6    2    4    0
C =   122
D =   122
```

2．数组的关系运算

在 MATLAB 中提供了 6 种数组关系运算符，即<（小于）、<=（小于等于）、>（大于）、>=（大于等于）、==（恒等于）、~=（不等于）。

关系运算的运算法则如下：

- 当两个比较量是标量时，直接比较两个数的大小。若关系成立，则返回的结果为 1，否则为 0。
- 当两个比较量是维数相等的数组时，逐一比较两个数组相同位置的元素，并给出比较结果。最终的关系运算结果是一个与参与比较的数组维数相同的数组，其组成元素为 0 或 1。

例 3-12，数组的关系运算示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 5 6 8 9 6]
B=[9 5 6 2 4 0]
C=A<6          %数组与常数比较，小于
D=A>=6         %数组与常数比较，大于等于
E=A<B          %数组与数组比较，小于
F=A==B         %数组与数组比较，恒等于
```

输出结果：

```
A =    1    5    6    8    9    6
B =    9    5    6    2    4    0
C =    1    1    0    0    0    0
D =    0    0    1    1    1    1
E =    1    0    0    0    0    0
F =    0    1    1    0    0    0
```

3．数组的逻辑运算

在 MATLAB 中数组提供了 3 种数组逻辑运算符，即&（与）、|（或）和~（非）。逻辑运算的运算法则如下：

- 如果是非零元素则为真，用 1 表示；如果是零元素则为假，用 0 表示。
- 当两个比较量是维数相等的数组时，逐一比较两个数组相同位置的元素，并给出

比较结果。最终的关系运算结果是一个与参与比较的数组维数相同的数组，其组成元素为 0 或 1。

- 与运算（*a*&*b*）时，*a*、*b* 全为非零，则为真，运算结果为 1；或运算（*a*|*b*）时，只要 *a*、*b* 有一个为非零，则运算结果为 1；非运算（~*a*）时，若 *a* 为 0，运算结果为 1，*a* 为非零，运算结果为 0。

例 3-13，数组的逻辑运算示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 5 6 8 9 6]
B=[9 5 6 2 4 0]
C=A&B %与
D=A|B %或
E=~B  %非
```

输出结果：

```
A =    1    5    6    8    9    6
B =    9    5    6    2    4    0
C =    1    1    1    1    1    0
D =    1    1    1    1    1    1
E =    0    0    0    0    0    1
```

3.2　矩阵操作

3.2.1　创建矩阵

除了在之前章节提到的零矩阵、单位矩阵和全 1 矩阵等特殊矩阵，MATLAB 中还有一些指令用于生成试验矩阵。

表 3-1 给出了 MATLAB 中其他特殊矩阵的指令集。

表 3-1　MATLAB 中特殊矩阵的指令集

函数名称	表示意义
compan(*p*)	生成一个特征多项式为 *p* 的二维矩阵
hadamard(*k*)	返回一个阶数为 *n*=2*k* 的 Hadamard 矩阵，只有当 *n* 能被 4 整除时 Hadamard 矩阵才存在
hankel(*x*)	返回一个由向量 ***x*** 定义的 Hankel 方阵。该矩阵是一个对称矩阵，其元素为 $h_{ij}=x_{i+j-a}$，第 1 列为向量 ***x***，反三角以下的元素为 0
hankel(*x*,*y*)	返回一个 *m*×*n* 的 Hankel 矩阵，它的第 1 列为向量 ***x***，最后一行为向量 ***y***
magic(*n*)	返回一个 *n*×*n* 的魔方矩阵
pascal(*n*)	返回一个 *n*×*n* 的 Pascal 矩阵
rosser	给出 Rosser 矩阵，这是一个经典对称特征测试问题，它的大小是 8×8
vander(*x*)	返回一个 Vandermonde 矩阵，其元素为 $v_{i,j}=x_i^{n-j}$，*n* 为向量 ***x*** 的长度
wilkinson(*n*)	返回一个 *m*×*n* 的 Wilkinson 特征值测试矩阵

1．希尔伯特（Hilbert）矩阵

希尔伯特（Hilbert）矩阵，也称 ***H*** 阵，其元素为 $H_{ij}=1/(i+j-1)$。由于它是一个条件数差的矩阵，所以将它用来作为试验矩阵。

关于希尔伯特矩阵的指令函数如下：

- hilb(n)：用于生成一个 $n \times n$ 的希尔伯特矩阵。
- invhilb(n)：用于生成一个 $n \times n$ 的希尔伯特矩阵的逆矩阵整数矩阵。

例 3-14，希尔伯特矩阵生成示例。

在命令行窗口中输入：

```
A=hilb(3)
```

输出结果：

```
A =
   1.0000   0.5000   0.3333
   0.5000   0.3333   0.2500
   0.3333   0.2500   0.2000
```

在命令行窗口中输入：

```
B=invhilb(3)
```

输出结果：

```
B =
    9   -36    30
  -36   192  -180
   30  -180   180
```

从结果中可以看出，希尔伯特矩阵和它的逆矩阵都是对称矩阵。

2．托普利兹（Toeplitz）矩阵

另外一个比较重要的矩阵为托普利兹（Toeplitz）矩阵，它由两个向量定义，一个行向量和一个列向量。对称的托普利兹矩阵由单一向量来定义。

关于托普利兹矩阵的指令函数如下：

- toeplitz(k,r)：用于生成非对称托普利兹矩阵，第 1 列为 ***k***，第 1 行为 ***r***，其余元素等于其左上角元素。
- toeplitz(c)：用于用向量 ***c*** 生成一个对称的托普利兹矩阵。

例 3-15，托普利兹矩阵函数示例。

输入命令：

```
C=toeplitz(2:5,2:2:8)
```

输出结果：

```
C =
     2     4     6     8
     3     2     4     6
     4     3     2     4
     5     4     3     2
```

3．0～1 间均匀分布的随机矩阵

在 MATLAB 中常用 rand()函数产生 0～1 间均匀分布的随机矩阵，其调用格式如下：

- r = rand(n)：产生维数为 $n \times n$ 的 0～1 间均匀分布的随机矩阵。
- r = rand(m,n)：产生维数为 $n \times m$ 的 0～1 间均匀分布的随机矩阵。
- r = rand(m,n,p,...)：产生维数为 $n \times m \times p$ 的 0～1 间均匀分布的随机矩阵。
- r = rand(size(A))：产生维数为 $n \times m \times p$ 与矩阵 $\boldsymbol{A}$ 相同的 0～1 间均匀分布的随机矩阵。

例 3-16，创建 0～1 间均匀分布的随机矩阵示例。

在命令行窗口中输入如下语句：

```
clear all
B=rand(3)
C=rand([3,4])
D=rand(size(C))
```

输出结果：

```
B =
   0.4218   0.9595   0.8491
   0.9157   0.6557   0.9340
   0.7922   0.0357   0.6787
C =
   0.7577   0.6555   0.0318   0.0971
   0.7431   0.1712   0.2769   0.8235
   0.3922   0.7060   0.0462   0.6948
D =
   0.3171   0.4387   0.7952   0.4456
   0.9502   0.3816   0.1869   0.6463
0.0344   0.7655   0.4898   0.7094
```

4. 标准正态分布随机矩阵

在 MATLAB 中常用 randn()函数产生均值为 0、方差为 1 的随机矩阵，其调用格式如下：

- r = randn(n)。
- r = randn(m,n)。
- r = randn(m,n,p,...)。
- r = randn([m,n,p,...])。
- r = randn(size(A))。

其格式可参考上述 rand()函数。

例 3-17，创建标准正态分布随机矩阵示例。

在命令行窗口中输入如下语句：

```
clear all
B=randn(3)
C=randn([3,4])
D=randn(size(C))
```

输出结果：

```
B =
   1.1093  -1.2141   1.5326
  -0.8637  -1.1135  -0.7697
   0.0774  -0.0068   0.3714
C =
  -0.2256   0.0326   1.5442  -0.7423
   1.1174   0.5525   0.0859  -1.0616
```

```
   -1.0891   1.1006  -1.4916   2.3505
D =
   -0.6156   0.8886  -1.4224  -0.1961
    0.7481  -0.7648   0.4882   1.4193
   -0.1924  -1.4023  -0.1774   0.2916
```

5．魔方矩阵

在 MATLAB 中常用 magic()函数产生魔方矩阵。魔方矩阵中每行、列和两条对角线上的元素和相等，其调用格式如下：

- M= magic(n)。

例 3-18，创建魔方矩阵示例。

在命令行窗口中输入如下语句：

```
clear all
A=magic(3)
B=magic(4)
C=magic(5)
E=sum(A)   %计算每行的和
F=sum(A')  %计算每列的和
```

输出结果：

```
A =
     8     1     6
     3     5     7
     4     9     2
B =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
C =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
E =    15    15    15
F =    15    15    15
```

6．帕斯卡矩阵

在 MATLAB 中常用 pascal()函数产生帕斯卡矩阵，其调用格式如下：

- A=pascal(n) ：返回 *n* 阶的对称正定 Pascal 矩阵，其中的元素是由 Pascal 三角组成的，其逆矩阵的元素都是整数。
- A = pascal(n,1)：返回由下三角的 Cholesky 因子组成的 Pascal 矩阵，它是对称的，所以它是自己的逆。
- A = pascal(n,2)：返回 pascal(n,1)的转置和交换形式。*A* 是单位矩阵的立方根。

例 3-19，创建帕斯卡矩阵示例。

在命令行窗口中输入如下语句：

```
clear all
A=pascal(4)    %创建 4 阶帕斯卡矩阵
```

```
B= pascal(3,2)
```

输出结果：

```
A =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
B =
     1      1      1
      -2    -1     0
     1      0      0
```

7．范德蒙（Vandermonde）矩阵

在 MATLAB 中常用 vander()函数产生范德蒙矩阵，其调用格式如下：

- A = vander(v)：生成范德蒙矩阵，矩阵的列是向量 *v* 的幂，即 *A*(*i*,*j*)=*v*(*i*)^(*n*-*j*)，其中 *n*=length (*v*)。

例 3-20，生成范德蒙（Vandermonde）矩阵示例。

在命令行窗口中输入如下语句：

```
clear all
A=vander([1 2 3 4])
B=vander([1;2;3;4])
C=vander(1:.5:3)
```

输出结果：

```
A =
     1     1     1     1
     8     4     2     1
    27     9     3     1
    64    16     4     1
B =
     1     1     1     1
     8     4     2     1
    27     9     3     1
    64    16     4     1
C =
    1.0000    1.0000    1.0000    1.0000    1.0000
    5.0625    3.3750    2.2500    1.5000    1.0000
   16.0000    8.0000    4.0000    2.0000    1.0000
   39.0625   15.6250    6.2500    2.5000    1.0000
   81.0000   27.0000    9.0000    3.0000    1.0000
```

> **注意**
>
> vander()函数产生范德蒙矩阵，输入向量可以使行向量或列向量。

3.2.2 改变矩阵大小

1．矩阵的合并

矩阵的合并就是把两个或者两个以上的矩阵数据连接起来得到一个新的矩阵。

注意

前面介绍的矩阵构造符[]不仅可用于构造矩阵，同时还可以作为一个矩阵合并操作符。表达式 ***C***=[***A B***]在水平方向合并矩阵 ***A*** 和 ***B***，而表达式 ***C***=[***A***;***B***]在竖直方向合并矩阵 ***A*** 和 ***B***。

例 3-21，合并矩阵示例一。

在命令行窗口中输入如下语句：

```
C=toeplitz(2:5,2:2:8)
```

输出结果：

```
C =
    2    4    6    8
    3    2    4    6
    4    3    2    4
    5    4    3    2
```

例 3-22，合并矩阵示例二。

在命令行窗口中输入如下语句：

```
clear all
a=ones(3,4)
b=zeros(3,4)
c=[a;b]
```

输出结果：

```
a =
    1    1    1    1
    1    1    1    1
    1    1    1    1
b =
    0    0    0    0
    0    0    0    0
    0    0    0    0
c =
    1    1    1    1
    1    1    1    1
    1    1    1    1
    0    0    0    0
    0    0    0    0
    0    0    0    0
```

例 3-23，合并矩阵示例三。

在命令行窗口中输入如下语句：

```
clear all
a=ones(3,4);
b=zeros(3,4);
c=[a b]
```

输出结果：

```
c =
    1    1    1    1    0    0    0    0
    1    1    1    1    0    0    0    0
    1    1    1    1    0    0    0    0
```

可以用矩阵合并符来构造任意大小的矩阵。不过需要注意的是，在矩阵合并的过程中一定要保持矩阵的形状是方形，否则矩阵合并将无法进行。

图 3-1 表明具有相同高度的两个矩阵可以在水平方向合并为一个新的矩阵。而图 3-2 则表明不具有相同高度的两个矩阵，不允许合并为一个矩阵。

图 3-1　正确的矩阵合并

图 3-2　不正确的矩阵合并

除了使用矩阵合并符“[]”来合并矩阵，还可以使用矩阵合并函数来合并矩阵。这些矩阵合并函数的函数描述和基本调用格式如表 3-1 所示。

2．矩阵行列的删除

要删除矩阵的某一行或者某一列，只要把该行或者该列赋予一个空矩阵“[]”即可。例如有一个 4×4 的随机矩阵，代码设置如下：

```
A=rand(4,4)
```

上述语句得到矩阵 $\boldsymbol{A}$：

```
A=
0.8147   0.6324   0.9575   0.9572
   0.9058   0.0975   0.9649   0.4854
   0.1270   0.2785   0.1576   0.8003
   0.9134   0.5469   0.9706   0.1419
```

如果想删除矩阵的第 2 行，则可以用如下语句：

```
A(2,:)=[]
```

由上述语句得到新的矩阵 $\boldsymbol{A}$：

```
A=
   0.8147   0.6324   0.9575   0.9572
   0.1270   0.2785   0.1576   0.8003
   0.9134   0.5469   0.9706   0.1419
```

3.2.3　重构矩阵

矩阵重构的两个比较重要的运算是转置和共轭转置，在 MATLAB 中，用在函数后面加撇号“'”来表示。在线性代数的专业书籍中，这种运算经常用*和 H 表示。

如果 $\boldsymbol{A}$ 是一个实数矩阵，那么它被转置时，第 1 行变成第 1 列，第 2 行变成第 2 列……以此类推，一个 $m\times n$ 矩阵变为一个 $n\times m$ 矩阵。如果矩阵是方阵，那么这个矩阵在主对角

线上反映出来。如果矩阵 $\boldsymbol{A}$ 的元素 a_{ij} 是复数，那么所有元素也是共轭的。如果仅希望转置，在撇号“'”之前输入一点，即成为“.'”，**A**.'表示转置，其结果与 conj(**A**')相同。如果 **A** 是实数矩阵，那么 **A**'与 **A**.'相同。

例 3-24，矩阵的重构示例。

在命令行窗口中输入如下语句：

```
C=toeplitz(2:5,2:2:8);
C=C'
```

输出结果：

```
C =
     2     3     4     5
     4     2     3     4
     6     4     2     3
     8     6     4     2
```

3.3　矩阵元素的运算

矩阵的加、减、乘、除、比较运算和逻辑运算等代数运算是 MATLAB 数值计算最基础的部分。本节将重点介绍这些运算。

3.3.1　矩阵的加减运算

进行矩阵加法、减法运算的前提是参与运算的两个矩阵或多个矩阵必须具有相同的行数和列数，即 **A**、**B**、**C** 等多个矩阵均为 $m \times n$ 矩阵；或者其中有一个或多个矩阵为标量。

在上述前提下，对于同型的两个矩阵，其加减法定义如下：

$\boldsymbol{C}=\boldsymbol{A}\pm\boldsymbol{B}$，矩阵 $\boldsymbol{C}$ 的各元素 $\boldsymbol{C}_{mn}=\boldsymbol{A}_{mn}+\boldsymbol{B}_{mn}$。

当其中含有标量 $\boldsymbol{x}$ 时，

$\boldsymbol{C}=\boldsymbol{A}\pm\boldsymbol{x}$，矩阵 $\boldsymbol{C}$ 的各元素 $\boldsymbol{C}_{mn}=\boldsymbol{A}_{mn}+\boldsymbol{x}$。

由于矩阵的加法运算归结为其元素的加法运算，容易验证，因此矩阵的加法运算满足下列运算律。

（1）交换律：$\boldsymbol{A}+\boldsymbol{B}=\boldsymbol{B}+\boldsymbol{A}$。

（2）结合律：$\boldsymbol{A}+(\boldsymbol{B}+\boldsymbol{C})=(\boldsymbol{A}+\boldsymbol{B})+\boldsymbol{C}$。

（3）存在零元：$\boldsymbol{A}+\boldsymbol{0}=\boldsymbol{0}+\boldsymbol{A}=\boldsymbol{A}$。

（4）存在负元：$\boldsymbol{A}+(-\boldsymbol{A})=(-\boldsymbol{A})+\boldsymbol{A}$。

例 3-25，矩阵加减法运算示例。

已知矩阵 $\boldsymbol{A}=\begin{bmatrix} 10 & 5 & 79 & 4 & 2 \\ 1 & 0 & 66 & 8 & 2 \\ 4 & 6 & 1 & 1 & 1 \end{bmatrix}$，矩阵 $\boldsymbol{B}=\begin{bmatrix} 9 & 5 & 3 & 4 & 2 \\ 1 & 0 & 4 & -23 & 2 \\ 4 & 6 & -1 & 1 & 0 \end{bmatrix}$，行向量 $\boldsymbol{C}=\begin{bmatrix} 2 & 1 \end{bmatrix}$，标量 $\boldsymbol{x}=20$，试求 $\boldsymbol{A}+\boldsymbol{B}$、$\boldsymbol{A}-\boldsymbol{B}$、$\boldsymbol{A}+\boldsymbol{B}+\boldsymbol{x}$、$\boldsymbol{A}-\boldsymbol{x}$、$\boldsymbol{A}-\boldsymbol{C}$。

在命令行窗口中输入如下语句：

```
A = [10 5 79 4 2;1 0 66 8 2;4 6 1 1 1];
B = [9 5 3 4 2;1 0 4 -23 2;4 6 -1 1 0];
x = 20;
C = [2 1];
ApB= A+B
AmB= A-B
ApBpX= A+B+x
AmX= A-x
AmC= A-C
```

得到的结果为：

```
ApB =
    19    10    82     8     4
     2     0    70   -15     4
     8    12     0     2     1
AmB =
     1     0    76     0     0
     0     0    62    31     0
     0     0     2     0     1
ApBpX =
    39    30   102    28    24
    22    20    90     5    24
    28    32    20    22    21
AmX =
   -10   -15    59   -16   -18
   -19   -20    46   -12   -18
   -16   -14   -19   -19   -19
错误使用  -
矩阵维度必须一致。
```

在 ***A***-***C*** 的运算中，MATLAB 返回错误信息，并提示矩阵的维数必须相等。这也证明了矩阵进行加减法运算必须满足一定的前提条件。

3.3.2 矩阵的乘法运算

MATLAB 中矩阵的乘法运算包括两种：数与矩阵的乘法；矩阵与矩阵的乘法。

1．数与矩阵的乘法

由于单个数在 MATLAB 中是以标量来存储的，因此数与矩阵的乘法也可以称为标量与矩阵的乘法。

设 x 为一个数，$\boldsymbol{A}$ 为矩阵，则定义 x 与 $\boldsymbol{A}$ 的乘积 $\boldsymbol{C}=x\boldsymbol{A}$ 仍为一个矩阵，$\boldsymbol{C}$ 的元素就是用数 x 乘矩阵 $\boldsymbol{A}$ 中对应的元素而得到的，即 $\boldsymbol{C}_{mnx}=x\boldsymbol{A}_{mn}$。数与矩阵的乘法满足下列运算律：

- $1\boldsymbol{A}=\boldsymbol{A}$。
- $x(\boldsymbol{A}+\boldsymbol{B})=x\boldsymbol{A}+x\boldsymbol{B}$。
- $(x+y)\boldsymbol{A}=x\boldsymbol{A}+y\boldsymbol{A}$。
- $(xy)\boldsymbol{A}=x(y\boldsymbol{A})=y(x\boldsymbol{A})$。

例 3-26，矩阵数乘示例。

已知矩阵 $A=\begin{bmatrix}0 & 3 & 3\\1 & 1 & 0\\-1 & 2 & 3\end{bmatrix}$，$E$ 是 3 阶单位矩阵，$E=\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$，试求表达式 $2A+3E$。

输入命令：

```
A = [0 3 3;1 1 0;-1 2 3];
E = eye(3);
R=2*A+3*E
```

得到结果：

```
R =
     3     6     6
     2     5     0
    -2     4     9
```

2. 矩阵与矩阵的乘法

两个矩阵的乘法必须满足被乘矩阵的列数与乘矩阵的行数相等。设矩阵 A 为 $m\times h$ 矩阵，B 为 $h\times n$ 矩阵，则两矩阵的乘积 $C=A\times B$ 为一个矩阵，且 $C_{mn}=\sum_{h=1}^{H} A_{mh}\times B_{hn}$ 。

矩阵之间的乘法不遵循交换律，即 $A\times B\neq B\times A$。但矩阵乘法遵循下列运算律。

- 结合律：$(A\times B)\times C=A\times(B\times C)$。
- 左分配律：$A\times(B+C)=A\times B+A\times C$。
- 右分配律：$(B+C)\times A=B\times A+C\times A$。
- 单位矩阵的存在性：$E\times A=A$，$A\times E=A$。

例 3-27，矩阵乘法的示例。

已知矩阵 $A=\begin{bmatrix}2 & 1 & 4 & 0\\1 & -1 & 3 & 4\end{bmatrix}$，矩阵 $B=\begin{bmatrix}1 & 3 & 1\\0 & -1 & 2\\1 & -3 & 1\\4 & 0 & -2\end{bmatrix}$，试求矩阵乘积 AB 及 BA。

输入命令：

```
A = [2 1 4 0;1 -1 3 4];
B = [1 3 1;0 -1 2;1 -3 1;4 0 -2];
R1= A*B
R2= B*A
```

得到结果：

```
R1 =
     6    -7     8
    20    -5    -6
错误使用  *
内部矩阵维度必须一致。                % 由于不满足矩阵的乘法条件，故 BA 无法计算
```

3.3.3 矩阵的除法运算

矩阵的除法是乘法的逆运算，分为左除和右除两种，分别用运算符号“\”和“/”表示。

A*******B*** 表示矩阵 ***A*** 的逆乘以 ***B***，***A*****/*****B*** 表示矩阵 ***A*** 乘以矩阵 ***B*** 的逆。除非矩阵 ***A*** 和矩阵 ***B*** 相同，否则 ***A*****/*****B*** 和 ***A**********B*** 是不等价的。对于一般的二维矩阵 ***A*** 和 ***B***，当进行 ***A**********B*** 运算时，要求 ***A*** 的行数与 ***B*** 的行数相等；当进行 ***A/B*** 运算时，要求 ***A*** 的列数与 ***B*** 的列数相等。

例 3-28，矩阵除法的示例。

设矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 2\\1 & 3\end{bmatrix}$，矩阵 $\boldsymbol{B}=\begin{bmatrix}1 & 0\\1 & 2\end{bmatrix}$，试求 ***A**********B*** 和 ***A/B***。

输入命令：

```
A = [1 2;1 3];
B = [1 0;1 2];
R1=A\B
R2=A/B
```

得到结果：

```
R1 =
    1    -4
    0     2
R2 =
                 0   1.000000000000000
  -0.500000000000000   1.500000000000000
```

3.3.4　矩阵的幂运算

当矩阵 ***A*** 为方阵时，可进行矩阵的幂运算，其定义为：$C = A^n = \underbrace{A \times A \times \cdots \times A}_{n}$。在 MATLAB 中，使用运算符号“^”表示幂运算。

例 3-29，方阵幂运算示例。

对于给定方阵 $\boldsymbol{A}=\begin{bmatrix}2 & 3 & 8\\3 & 1 & -1\\5 & 0 & 4\end{bmatrix}$，试求 $\boldsymbol{A}^2$、$\boldsymbol{A}^5$。

输入命令：

```
A = [2 3 8;3 1 -1;5 0 4];
A2= A^2
A5= A^5
```

得到结果：

```
A2 =
   53     9    45
    4    10    19
   30    15    56
A5 =
      37496      13827      52622
      10077       2887      11891
      34295      11250      44464
```

3.3.5　矩阵元素的查找

MATLAB 中函数 find()的作用是进行矩阵元素的查找，它通常与关系函数和逻辑运算相结合。其调用格式如下：

- ind = find(X)：该函数查找矩阵 ***X*** 中的非零元素，函数返回这些元素的单下标。
- [row,col] = find(X, ...)：该函数查找矩阵 ***X*** 中的非零元素，函数返回这些元素的双下标 ***i*** 和 ***j***。

例 3-30，利用函数 find()查找矩阵中的元素示例。

在命令行窗口中输入如下语句：

```
clear all;
A=[1 3 0;3 1 0;9 2 4];
B=find(A) ;                   %矩阵中非零元素的下标
C=find(A>=1);                 %矩阵中大于等于 1 的元素的下标
D=A(A>=1);                    %矩阵中大于等于 1 的元素
BCD=[B,C,D]
A(find(A==0))=10              %矩阵中等于 0 的元素改为 10
```

输出结果：

```
BCD =
     1     1     1
     2     2     3
     3     3     9
     4     4     3
     5     5     1
     6     6     2
     9     9     4
A =
     1     3    10
     3     1    10
     9     2     4
```

3.3.6　矩阵元素的排序

MATLAB 中函数 sort()的作用是按照升序排序，排序后的矩阵和原矩阵的维数相同。其调用格式如下：

- B = sort(A)：该函数对矩阵 ***A*** 进行升序排列。***A*** 可为矩阵或向量。
- B = sort(A,dim)：该函数对矩阵 ***A*** 进行升序排列，并将结果返回在给定的维数 dim 上按照升序排列。当 dim=1 时，按照列进行排序；当 dim=2 时，按照行进行排序。
- B = sort(...,mode)：该函数对矩阵 ***A*** 进行排序，mode 可指定排序的方式。ascend 指定按升序排列，为默认值；descend 指定按降序排列。

例 3-31，矩阵元素的排序示例。

在命令行窗口中输入如下语句：

```
clear all;
```

```
A=[1 3 0;3 1 0;9 2 4];
B=sort(A);                  %矩阵中元素按照列进行升序排序
C=sort(A,2);                %矩阵中元素按照行进行升序排序
D=sort(A,'descend');        %矩阵中元素按照列进行降序排序
E=sort(A,2,'descend');      %矩阵中元素按照行进行降序排序
BCDE=[B C;D E]
```

输出结果：

```
BCDE =
    1    1    0    0    1    3
    3    2    0    0    1    3
    9    3    4    2    4    9
    9    3    4    3    1    0
    3    2    0    3    1    0
    1    1    0    9    4    2
```

例 3-32，对向量进行排序示例。

在命令行窗口中输入如下语句：

```
A = [78 23 10 100 45 5 6];
sort(A)
```

输出结果：

```
ans =     5     6    10    23    45    78   100
```

3.3.7 矩阵元素的求和

MATLAB 中函数 sum()和 cumsum()的作用是对矩阵的元素求和。其调用格式如下：

- B = sum(A)：该函数对矩阵 ***A*** 的元素求和，返回由矩阵 ***A*** 各列元素的和组成的向量。
- B = sum(A,dim)：该函数返回在给定的维数 dim 上元素的和。当 dim=1 时，计算矩阵 ***A*** 各列元素的和；当 dim=2 时，计算矩阵 ***A*** 各行元素的和。
- B = cumsum(A)。
- B = cumsum(A,dim)。

函数 cumsum()的调用格式与 sum()类似，不同的是其返回值为矩阵。下面通过示例查看两个函数的不同之处。

例 3-33，矩阵的求和示例。

在命令行窗口中输入如下语句：

```
clear all;
A=[1 3 0;3 1 0;9 2 4];
B=sum(A)                    %矩阵中元素按照列进行求和
C=sum(A,2)                  %矩阵中元素按照行进行求和
D=cumsum(A)                 %矩阵中各列元素的和
E=cumsum(A,2)               %矩阵中各行元素的和
F=sum(sum(A))               %矩阵中所有元素的和
```

输出结果：

```
B =   13    6    4
C =
    4
    4
```

```
    15
D =
     1     3     0
4     4     0
    13     6     4
E =
     1     4     4
     3     4     4
     9    11    15
F =  23
```

提示

通过 sum(sum())可求出矩阵所有元素的和。

3.3.8　矩阵元素的求积

MATLAB 中函数 prod()和 cumprod()的作用是对矩阵的元素求积。其调用格式如下：

- B = prod(A)：该函数对矩阵 ***A*** 的元素求积，返回由矩阵 ***A*** 各列元素的积组成的向量。
- B = prod(A,dim)：该函数返回在给定的维数 dim 上元素的积。当 dim=1 时，计算矩阵 ***A*** 各列元素的积；当 dim=2 时，计算矩阵 ***A*** 各行元素的积。
- B = cumprod(A)。
- B = cumprod(A,dim)。

函数 cumprod()的调用格式与 prod()类似，不同的是其返回值为矩阵。读者可以通过下面的示例查看两者的不同之处。

例 3-34，矩阵元素的求积示例。

在命令行窗口中输入如下语句：

```
clear all;
A= magic(3)
B=prod(A)                   %矩阵各列元素的积
C=prod(A,2)                 %矩阵各行元素的积
D=cumprod(A)                %矩阵各列元素的积
E=cumprod(A,2)              %矩阵各行元素的积
```

输出结果：

```
A =
     8     1     6
     3     5     7
     4     9     2
B =    96    45    84
C =
    48
   105
    72
D =
     8     1     6
    24     5    42
    96    45    84
```

```
E =
     8     8    48
     3    15   105
     4    36    72
```

3.3.9 矩阵元素的差分

MATLAB 中函数 diff()的作用是计算矩阵的差分。其调用格式如下：

- Y = diff(X)：计算矩阵各列元素的差分。
- Y = diff(X,n)：计算矩阵各列元素的 *n* 阶差分。
- Y = diff(X,n,dim)：计算矩阵在给定的维数 dim 上元素的 *n* 阶差分。当 dim=1 时，计算矩阵各列元素的差分；当 dim=2 时，计算矩阵各行元素的差分。

例 3-35，矩阵的差分计算示例。

在命令行窗口中输入如下语句：

```
clear all;
A= magic(3);
B=diff(A)               %矩阵各列元素的差分
C=diff(A,2)             %矩阵各列元素的 2 阶差分
D=diff(A,1,1)           %矩阵各列元素的差分
E=diff(A,1,2)           %矩阵各行元素的差分
```

输出结果：

```
B =
    -5     4     1
     1     4    -5
C =
     6     0    -6
D =
    -5     4     1
     1     4    -5
E =
    -7     5
     2     2
     5    -7
```

○ 提示

当参数 $n \geqslant$ size(x,dim)时，函数的返回值是空矩阵。

3.4 矩阵运算

矩阵运算是线性代数中极其重要的部分。本节将介绍 MATLAB 中与矩阵运算相关的内容，包括矩阵分析、特征值求解和奇异值等。

3.4.1　矩阵分析

MATLAB 提供的矩阵分析函数如表 3-2 所示。

表 3-2　矩阵分析函数

函 数 名	功能描述
norm	求矩阵或者向量的范数
normest	估计矩阵的 2 阶范数
rank	矩阵的秩，即求对角元素的和
det	矩阵的行列式
trace	矩阵的迹
null	0 空间
orth	正交化空间
rref	约化行阶梯形式
subspace	求两个矩阵空间的角度

1．向量和矩阵的范数运算

对于线性空间中的一个向量 $\boldsymbol{x}=\{x1,x2,\cdots,xn\}$，如果存在一个函数 $r(x)$满足以下 3 个条件：

（1）$r(\boldsymbol{x})>0$，且 $r(\boldsymbol{x})=0$ 的充要条件为 $\boldsymbol{x}=0$。

（2）$r(a\boldsymbol{x})=|a|\,r(\boldsymbol{x})$，其中 a 为任意标量。

（3）对向量 $\boldsymbol{x}$ 和 $\boldsymbol{y}$，有 $r(\boldsymbol{x}+\boldsymbol{y})\leqslant r(\boldsymbol{x})+r(\boldsymbol{y})$。

则称 $r(\boldsymbol{x})$为向量 $\boldsymbol{x}$ 的范数，一般记为$\|\boldsymbol{x}\|$。范数的形式多种多样，下面式子中定义的范数操作就满足以上 3 个条件：

$$\|\boldsymbol{x}\|_p=\left(\sum_{i=1}^{n}|x_t|^p\right)^{1/p},\ p=1,2,\cdots \text{ 且 } \|\boldsymbol{x}\|_\infty=\max_{1\le i\le n}|x_t|;\quad \|\boldsymbol{x}\|_{-\infty}=\min_{1\le i\le n}|x_t|$$

式中定义的 $\|\boldsymbol{x}\|_p$ 称为 p 阶范数，其中最有用的是 1、2 和 ∞ 阶范数。

矩阵的范数是基于向量的范数定义的，其定义式如下：

$$\|\boldsymbol{A}\|=\max_{x\neq 0}\frac{\|\boldsymbol{A}x\|}{\|x\|}$$

与向量的范数一样，矩阵的范数最常用的也是 1、2 和 ∞ 阶范数，它们的定义式如下：

$$\|\boldsymbol{A}\|_1=\max_{1\le j\le n}\sum_{j=1}^{n}|a_{ij}|,\ \|\boldsymbol{A}\|_2=\sqrt{S_{\max}\{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\}} \text{ 和 } \|\boldsymbol{A}\|_\infty=\max_{1\le t\le n}\sum_{j=1}^{n}|a_{ij}|$$

在上式中，$S_{\max}\{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\}$则为矩阵 $\boldsymbol{A}$ 的最大奇异值的平方。

在 MATLAB 中，求向量范数的函数具体用法如下：

- N=norm(x,p)：对任意大于 1 的 ***p*** 值，返回向量 ***x*** 的 ***p*** 阶范数。
- N=norm(x)：返回向量的 2 阶范数，相当于 ***N***=norm(***x***,2)。
- N=norm(x,inf)：返回向量的¥阶范数，相当于 ***N***=max(abs(***x***))。
- N=norm(x,-inf)：返回向量的-¥阶范数，相当于 ***N***=min(abs(***x***))。

在 MATLAB 中，求矩阵范数的函数具体用法如下：

- N=norm(A)：计算矩阵的 2 阶范数，也就是最大奇异值。
- N=norm(A,p)：根据参数 ***p*** 的值不同，求不同阶的范数值。当 ***p***=1 时，计算矩阵 ***A*** 的 1 阶范数，相当于 max(sum(abs(***A***)))。当 ***p***=2 时，计算矩阵 ***A*** 的 2 阶范数，相当于 norm(***A***)。当 ***p***=inf 时，计算矩阵 ***A*** 的¥阶范数，相当于 max(sum(abs(***A***')))。当 ***p***=pro 时，计算矩阵 ***A*** 的 F 范数（Frobenius 范数），相当于 sqrt(sum(diag(***A***'****A***)))。

例 3-36，求向量 ***x*** 的 2 阶范数示例。

在命令行窗口中输入如下语句：

```
norm(1:6,2)
```

输出结果：

```
ans =9.5394
```

○ 注意

当矩阵维数比较大时，会导致计算矩阵范数的时间比较长，并且当一个近似的范数值满足要求时，可以考虑使用函数 normest()来估计 2 阶范数值。函数 normest()最初开发时是为了提供给稀疏矩阵使用的，同时它也能接收满矩阵的输入，一般在满矩阵维数比较大时使用。

函数 normest()的用法如下：

- normest(S)：估计矩阵 ***S*** 的 2 阶范数值。
- normest(S,tol)：使用 tol 作为允许的相对误差。

例 3-37，求矩阵的范式示例。

在命令行窗口中输入如下语句：

```
clear all
a=[1 2 3;3 4 5;7 8 9];
b=norm(a,1);            %矩阵的 1 阶范式
c=norm(a) ;             %矩阵的 2 阶范式
d=norm(a,inf) ;         %矩阵的无穷范式
e=norm(a,'fro');        %矩阵的 Frobenius 范式
f=normest(a);           %矩阵的 2 阶范式的估计值
bcdef=[ b c d e f ]
```

输出结果：

```
bcdef =  17  16.02158  24  16.06238  16.02158
```

2．矩阵的秩

矩阵 ***A*** 中线性无关的列向量个数称为列秩，线性无关的行向量个数称为行秩。MATLAB 中用函数 rank()来计算矩阵的秩。函数 rank()的用法如下：

- rank(A)：用默认允许误差计算矩阵的秩。

- rank(A,tol)：给定允许误差计算矩阵的秩，tol = max(size(***A***))・eps(norm(***A***))。

例 3-38，求矩阵的秩的示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 2 3;3 4 5;7 8 9];
B=magic(3);
r1=rank(A)                %矩阵的秩
r2=rank(B)
```

输出结果：

```
r1 =    2
r2 =    3
```

3．矩阵的行列式

矩阵 $\boldsymbol{A}=\{a_{ij}\}_{n\times n}$ 的行列式定义如下：

$$\|\boldsymbol{A}\| = \det(\boldsymbol{A}) = \sum_{k=1}^{n} (-1)^{k} a_{1}k_{1}a_{2}k_{2} \cdots a_{n}k_{n}$$

其中，$k_1,k_2,\cdots,k_n$ 是将序列 $1,2,\cdots,n$ 交换 k 次所得的序列。在 MATLAB 中用函数 det()来计算矩阵的行列式。

例 3-39，计算矩阵的行列式示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 2 3;3 4 5;7 8 9];
B=magic(5);
r1=det(A);                %矩阵的行列式
r2=det(B);
disp(['a 的行列式值=',num2str(r1)]);
disp(['b 的行列式值=',num2str(r2)]);
```

输出结果：

```
A 的行列式值=0
B 的行列式值=5070000
```

4．矩阵的迹

矩阵的迹定义为矩阵对角元素之和。在 MATLAB 中用函数 trace()来计算矩阵的迹。

例 3-40，矩阵的迹计算示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 2 3;3 4 5;7 8 9];
B=magic(5);
r1=trace(A);                  %矩阵的迹
r2=trace(B);
disp(['A 的迹=',num2str(r1)]);
disp(['B 的迹=',num2str(r2)]);
```

输出结果：

```
A 的迹=14
B 的迹=65
```

5．矩阵的化零矩阵

MATLAB 中提供了求化零矩阵的函数 null()。其用法如下：

- Z = null(A)：返回矩阵 ***A*** 的一个化零矩阵，如果化零矩阵不存在则返回空矩阵。
- Z = null(A, 'r')：返回有理数形式的化零矩阵。

例 3-41，求矩阵的化零矩阵。

在命令行窗口中输入如下语句：

```
clear all
A=[1 2 3;3 4 5;7 8 9];
Z=null(A)                    %求矩阵 A 的化零矩阵
AZ=A*Z
ZR=null(A,'r')               %求矩阵 A 的有理数形式的化零矩阵
AZR=A*ZR
```

输出结果：

```
Z =
   0.4082
  -0.8165
   0.4082
AZ =
    -0.2220*1.0e-15
        0
        0
ZR =
    1
   -2
    1
AZR =
    0
    0
    0
```

6．矩阵的正交空间

矩阵 ***A*** 的正交空间 ***Q*** 具有 ***Q***'·***Q***=**I** 的性质，并且 ***Q*** 的列矢量构成的线性空间与矩阵 ***A*** 的列矢量构成的线性空间相同，且正交空间 ***Q*** 与矩阵 ***A*** 具有相同的秩。MATLAB 中提供了函数 orth()来求正交空间 ***Q***。

例 3-42，矩阵的正交空间求解示例。

在命令行窗口中输入如下语句：

```
clear all
A=[1 2 3;3 4 5;7 8 9];
Q=orth(A)
```

输出结果：

```
Q =
  -0.2262   -0.8143
  -0.4404   -0.4040
  -0.8688    0.4168
```

7．矩阵的约化行阶梯形式

矩阵的约化行阶梯形式是高斯-约旦消去法解线性方程组的结果，其形式为：

$$\begin{pmatrix} 1 & \mathrm{K} & 0 & * \\ \mathrm{M} & \mathrm{O} & \mathrm{M} & * \\ 0 & \mathrm{L} & 1 & * \end{pmatrix}$$

MATLAB 中提供了函数 rref()来求矩阵的约化行阶梯形式。其用法如下：

- R = rref(A)：返回矩阵 ***A*** 的约化行阶梯形式 ***R***。
- [R,jb] = rref(A)：返回矩阵 ***A*** 的约化行阶梯形式 ***R***，并返回 1×*r* 的向量 ***jb***，*r* 为矩阵 ***A*** 的秩；***A***(:,***jb***)是矩阵 ***A*** 的列矢量构成的线性空间；***R***(1:*r*,*jb*)是 *r*×*r* 的单位矩阵。
- [R,jb] = rref(A,tol)：以 tol 作为允许的相对误差计算矩阵 ***A*** 的秩。

例 3-43，求矩阵 ***A*** 的约化行阶梯形式示例。

在命令行窗口中输入如下语句：

```
A=[1 2 3;4 5 6;7 8 9;10 11 12];    %矩阵 A
R=rref(A)                          %正交矩阵 A 的约化行阶梯形式
```

输出结果：

```
R =
    1    0   -1
    0    1    2
    0    0    0
    0    0    0
```

8．矩阵空间之间的夹角

矩阵空间之间的夹角代表两个矩阵线性相关的程度。如果夹角很小，它们之间的线性相关度就很高；反之，它们之间的线性相关度就不大。在 MATLAB 中用函数 subspace()来实现求矩阵空间之间的夹角。其调用格式如下：

- theta = subspace(A,B)：返回矩阵 ***A*** 和矩阵 ***B*** 之间的夹角。

例 3-44，求矩阵 ***A*** 和 ***B*** 之间的夹角。

在命令行窗口中输入如下语句：

```
clear all
A=[1 2 3;3 4 5;7 8 9;8 7 9;0 2 8];
B=magic(5);
subspace(A,B)
```

输出结果：

```
ans =  5.7539e-16
```

3.4.2　矩阵分解

矩阵分解是把一个矩阵分解成几个“较简单”的矩阵连乘的形式。无论在理论上还是工程应用上，矩阵分解都是十分重要的。本节将介绍几种矩阵分解的方法，相关函数如表 3-3 所示。

表 3-3　矩阵分解函数

函　　数	功 能 描 述
chol	Cholesky 分解
cholinc	稀疏矩阵的不完全 Cholesky 分解
lu	矩阵 LU 分解
luinc	稀疏矩阵的不完全 LU 分解
qr	正交三角分解
svd	奇异值分解
gsvd	一般奇异值分解
schur	舒尔分解

在 MATLAB 中，线性方程组的求解主要基于 4 种基本的矩阵分解，即对称正定矩阵的 Cholesky 分解、一般方阵的高斯消去法分解、舒尔分解和矩形矩阵的正交分解。

1．对称正定矩阵的 Cholesky 分解

Cholesky 分解在 MATLAB 中用函数 chol()来实现，其常用的调用方式如下：

- R = chol(X)：其中 $\boldsymbol{X}$ 为对称正定矩阵，$\boldsymbol{R}$ 是上三角矩阵，使得 $\boldsymbol{X}=\boldsymbol{R}'\cdot\boldsymbol{R}$。如果 $\boldsymbol{X}$ 是非正定的，则结果将返回出错信息。
- [R,p] = chol(X)：返回两个参数，并且不会返回出错信息。当 $\boldsymbol{X}$ 是正定矩阵时，返回的上三角矩阵 $\boldsymbol{R}$ 满足 $\boldsymbol{X}=\boldsymbol{R}'\cdot\boldsymbol{R}$，且 $p=0$；当 $\boldsymbol{X}$ 是非正定矩阵时，返回值 p 是正整数，$\boldsymbol{R}$ 是上三角矩阵，其阶数为 p-1，且满足 $\boldsymbol{X}(1{:}p-1,1{:}p-1)=\boldsymbol{R}'\cdot\boldsymbol{R}$。

考虑线性方程组 $\boldsymbol{A}x=b$，其中 $\boldsymbol{A}$ 可以做 Cholesky 分解，使得 $\boldsymbol{A}=\boldsymbol{R}'\cdot\boldsymbol{R}$，这样线性方程组就可以改写成 $\boldsymbol{R}'\cdot\boldsymbol{R}\cdot x=b$，由于左除算符“\”可以快速处理三角矩阵，因此得出：

$$x=\boldsymbol{R}\backslash(\boldsymbol{R}'\backslash b)$$

如果 $\boldsymbol{A}$ 是 $n\times n$ 的方阵，则 chol($\boldsymbol{A}$)的计算复杂度是 $O(n^3)$，而左除算符“\”的计算复杂度只有 $O(n^2)$。

例 3-45，利用 chol 进行矩阵分解示例。

在命令行窗口中输入如下语句：

```
clear all;
A=pascal(5)%产生 5 阶帕斯卡矩阵
eig(A)
R=chol(A)
R'*R
```

输出结果：

```
A =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
ans =
    0.0108
    0.1812
```

```
   1.0000
   5.5175
  92.2904
R =
     1     1     1     1     1
     0     1     2     3     4
     0     0     1     3     6
     0     0     0     1     4
     0     0     0     0     1
ans =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
```

对于稀疏矩阵，MATLAB 提供了函数 cholinc()来做不完全 Cholesky 分解。函数 cholinc()的另一个优点是它可用于计算实半正定矩阵。函数 cholinc()的调用格式如下：

- R = cholinc(X,DROPTOL)：其中 ***X*** 和 ***R*** 的含义与函数 chol()中的变量相同，DROPTOL 为不完全 Cholesky 分解的丢失容限。当 DROPTOL 设为 0 时，退化为完全 Cholesky 分解。
- R = cholinc(X,OPTS)：其中 OPTS 为结构体，它有 3 个属性，即 DROPTOL、MICHOL 和 RDIAG。DROPTOL 为不完全 Cholesky 分解的丢失容限；MICHOL 为 1 时采用改进算法的不完全 Cholesky 分解，否则不采用改进算法；RDIAG 为 1 时 ***R*** 的对角元素中的零值替换成为 DROPTOL 的平方根，RDIAG 为 0 时不做此替换。
- R = cholinc(X, '0')：完全 Cholesky 分解。
- [R,p] = cholinc(X, '0')：返回两个参数，并且不会返回出错信息。当 ***X*** 是正定矩阵时，返回的上三角矩阵 ***R*** 满足 ***X***=***R***'·***R***，且 p=0；当 X 是非正定矩阵时，返回值 p 是正整数，***R*** 是上三角矩阵，其阶数为 p-1，且满足 ***X***(1:p-1,1:p-1)=***R***'· ***R***。
- R = cholinc(X, 'inf')：采用 Cholesky-Infinity 分解。Cholesky-Infinity 分解是基于 Cholesky 分解的，但它可以处理实半正定矩阵。

例 3-46，稀疏矩阵 Cholesky 分解示例。

在命令行窗口中输入如下语句：

```
clear all;
S = [2 0 3 0;0 25 0 30;3 0 91 0;0 32 0 661];
Rinf = cholinc(sparse(S),'inf');
Rinf=full(Rinf)
Rinf'* Rinf
```

输出结果：

```
Rinf =
  1.4142e+00          0   2.1213e+00          0
          0   5.0000e+00          0   6.0000e+00
          0          0   9.3005e+00          0
          0          0          0   2.5000e+01
ans =
```

```
   2.0000e+00          0   3.0000e+00          0
            0   2.5000e+01          0   3.0000e+01
   3.0000e+00          0   9.1000e+01          0
            0   3.0000e+01          0   6.6100e+02
```

2．一般方阵的高斯消去法分解

高斯消去法分解又称 LU 分解，它可以将任意一个方阵 ***A*** 分解为一个下三角矩阵 ***L*** 和一个上三角矩阵 ***U*** 的乘积，即 ***A***=***LU***。LU 分解在 MATLAB 中用函数 lu()来实现，其调用方式如下：

- [L,U] = lu(X)：***X*** 为一个方阵，***L*** 为“心理”下三角矩阵，***U*** 为上三角矩阵，满足关系 ***X***=***L***·***U***。
- [L,U,P] = lu(X)：***X*** 为一个方阵，***L*** 为下三角矩阵，***U*** 为上三角矩阵，***P*** 为置换矩阵，满足关系 ***P***·***X*** = ***L***·***U***。
- Y = lu(X)：***X*** 为一个方阵，把上三角矩阵和下三角矩阵合并在矩阵 ***Y*** 中给出，矩阵 ***Y*** 的对角元素为上三角矩阵的对角元素，即 ***Y***=***L***+***U***-***I***。置换矩阵 ***P*** 的信息丢失。

考虑线性方程组 ***A****x*=*b*，其中，对矩阵 ***A*** 可以做 LU 分解，使得 ***A***=***L***·***U***，这样线性方程组就可以改写成 ***L***·***U***·*x*=*b*，由于左除算运符“\”可以快速处理三角矩阵，因此可以快速解出：

$$x=\boldsymbol{U}\backslash(\boldsymbol{L}\backslash b)$$

利用 LU 分解来计算行列式的值和矩阵的逆，其命令形式如下：

- det(A)=det(L)*det(U)。
- inv(A)=inv(U)*inv(L)。

例 3-47，进行 LU 分解示例。

在命令行窗口中输入如下语句：

```
clear all;
A=[2 4 5;8 9 6;1 3 5];
[L1,U1]=lu(A)               %矩阵的 LU 分解
[L2,U2,P]=lu(A)
Y1=lu(A)
L1*U1                       %验证
Y2=L2+U2-eye(size(A))       %验证
```

输出结果：

```
L1 =
   0.2500   0.9333   1.0000
   1.0000        0        0
   0.1250   1.0000        0
U1 =
   8.0000   9.0000   6.0000
        0   1.8750   4.2500
        0        0  -0.4667
L2 =
   1.0000        0        0
   0.1250   1.0000        0
   0.2500   0.9333   1.0000
```

```
U2 =
   8.0000   9.0000   6.0000
        0   1.8750   4.2500
        0        0  -0.4667
P =
    0    1    0
    0    0    1
    1    0    0
Y1 =
   8.0000   9.0000   6.0000
   0.1250   1.8750   4.2500
   0.2500   0.9333  -0.4667
ans =
    2    4    5
    8    9    6
    1    3    5
Y2 =
   8.0000   9.0000   6.0000
   0.1250   1.8750   4.2500
0.2500   0.9333  -0.4667
```

此外，对于稀疏矩阵，MATLAB 提供了函数 luinc()来做不完全 LU 分解，其调用格式如下：

- [L U]= luinc(X,DROPTOL)：其中 ***X***、***L*** 和 ***U*** 的含义与函数 lu()中的变量相同，DROPTOL 为不完全 LU 分解的丢失容限。当 DROPTOL 设为 0 时，退化为完全 LU 分解。
- [L U] = luinc(X,OPTS)：其中 OPTS 为结构体，它有 4 个属性，即 DROPTOL、MICHOL、RDIAG 和 THRESH。DROPTOL 为不完全 LU 分解的丢失容限；MICHOL 为 1 时采用改进算法的不完全 LU 分解，否则不采用改进算法；RDIAG 为 1 时，***R*** 的对角元素中的零值替换成 DROPTOL 的平方根，为 0 时不做此替换；THRESH 是绕对角线旋转因子，其取值范围是[0,1]，当 THRESH 为 0 时强制绕对角线旋转，THRESH 的默认值是 1。
- [L,U,P] = luinc(X, '0')：0 级不完全 LU 分解。
- [L,U] = luinc(X, '0')：0 级不完全 LU 分解。
- Y = luinc(X, '0')：0 级完全 LU 分解。

3．矩形矩阵的正交分解

矩形矩阵的正交分解又称 QR 分解。QR 分解把一个 $m\times n$ 的矩阵 ***A*** 分解为一个正交矩阵 ***Q*** 和一个上三角矩阵 ***R*** 的乘积，即 ***A***=***Q***·***R***。在 MATLAB 中 QR 分解由函数 qr()来实现，下面介绍 QR 分解的调用方式。

- [Q,R] = qr(A)：其中矩阵 ***R*** 为与矩阵 ***A*** 具有相同大小的上三角矩阵，***Q*** 为正交矩阵，它们满足 ***A***=***Q***·***R***。该调用方式适用于满矩阵和稀疏矩阵。
- [Q,R] = qr(A,0)：为“经济”方式的 QR 分解。设矩阵 ***A*** 是一个 $m\times n$ 的矩阵，若 $m>n$，则只计算矩阵 ***Q*** 的前 n 列元素，***R*** 为 $n\times n$ 的矩阵；若 $m\leqslant n$，则与[***Q***,***R***] = qr(***A***)效果一致。该调用方式适用于满矩阵和稀疏矩阵。

- [Q,R,E] = qr(A)：***R*** 是上三角矩阵，***Q*** 为正交矩阵，***E*** 为置换矩阵。它们满足 ***A***·***E***=***Q***·***R***，程序选择一个合适的矩阵 ***E*** 使得 abs(diag(R))是降序排列的。该调用方式适用于满矩阵。
- [Q,R,E] = qr(A,0)：“经济”方式的 QR 分解，其中 ***E*** 是一个置换矢量。它们满足 ***A***(:,E)= ***Q***·***R***。该调用方式适用于满矩阵。
- R =qr(A)：返回上三角矩阵 ***R***，这里 ***R***= chol(***A***'·***A***)。该调用方式适用于稀疏矩阵。
- R = qr(A,0)：以“经济”方式返回上三角矩阵 ***R***。
- [C,R] = qr(A,B)：其中矩阵 ***B*** 必须与矩阵 ***A*** 具有相同的行数，矩阵 ***R*** 是上三角矩阵，***C***=***Q***' ·***B***。

例 3-48，通过 QR 分解分析矩阵的秩示例。

在命令行窗口中输入如下语句：

```
clear all;
A=[2 4 5;8 9 6;1 3 5];
[Q1,R1]=qr(A)
B=[2 4 5;8 9 6;1 3 5;5 4 10];
B_rank=rank(B);
disp(['矩阵B 的秩 = ',num2str(B_rank)]);
[Q2,R2]=qr(B)
Q1*R1
```

输出结果：

```
Q1 =
   -0.2408    0.6424   -0.7276
   -0.9631   -0.2511    0.0970
   -0.1204    0.7241    0.6791
R1 =
   -8.3066   -9.9920   -7.5843
         0    2.4818    5.3257
         0         0    0.3395
矩阵B 的秩 = 3
Q2 =
   -0.2063    0.5983   -0.2285    0.7398
   -0.8251    0.0774    0.5457   -0.1241
   -0.1031    0.6299   -0.3956   -0.6604
   -0.5157   -0.4892   -0.7025    0.0348
R2 =
   -9.6954  -10.6236  -11.6551
         0    3.0230    1.7138
         0         0   -6.8719
         0         0         0
ans =
    2.0000    4.0000    5.0000
    8.0000    9.0000    6.0000
    1.0000    3.0000    5.0000
```

4．舒尔分解

舒尔分解定义式为

$$A=U\cdot S\cdot U'$$

其中 ***A*** 必须是一个方阵，***U*** 是一个酉矩阵，***S*** 是一个块对角化矩阵，由对角线上的 1×1

和 2×2 块组成。特征值可以由矩阵 ***S*** 的对角块给出，而矩阵 ***U*** 给出比特征向量更多的数值特征。此外，对缺陷矩阵也可以进行舒尔分解。MATLAB 中用函数 schur()来进行舒尔分解，其调用格式如下：

- [U,S] = schur(A)：返回酉矩阵 ***U*** 和块对角矩阵 ***S***。
- S = schur(A)：仅返回块对角矩阵 ***S***。
- schur(A, 'real')：返回的实特征值放在对角线上，而把复特征值放在对角线上的 2×2 块中。
- schur(A, ' complex')：返回的矩阵 ***S*** 是上三角矩阵，并且如果矩阵 ***A*** 有复特征值，则矩阵 ***S*** 是复矩阵。

另外，函数 rsf2csf()可以把实数形式的舒尔矩阵转换成复数形式的舒尔矩阵。

例 3-49，舒尔分解示例。

在命令行窗口中输入如下语句：

```
clear all
A=pascal(5);
[U,S]=schur(A)
U*S*U'-A          %验证
```

输出结果：

```
U=
   0.1680  -0.5706  -0.7660   0.2429   0.0175
  -0.5517   0.5587  -0.3830   0.4808   0.0749
   0.7025   0.2529   0.1642   0.6110   0.2055
  -0.4071  -0.5179   0.4377   0.4130   0.4515
   0.0900   0.1734  -0.2189  -0.4074   0.8649
S=
   0.0108       0       0       0       0
        0   0.1812       0       0       0
        0       0   1.0000       0       0
        0       0       0     5.5175       0
        0       0       0       0      92.2904
ans =   1.0e-13 *
   0.0022   0.0022  -0.0022   0.0044       0
   0.0022  -0.0044       0   0.0178   0.0178
  -0.0022       0       0   0.0355       0
   0.0044   0.0089   0.0355   0.1421   0.1421
        0   0.0178       0   0.1421   0.2842
```

3.4.3　特征值和特征向量

1．特征值和特征向量的定义

MATLAB 中的命令计算特征值和特征向量十分方便，可以得到不同的子结果和分解，这在线性代数学习中十分有意义。本节中的命令只能对二维矩阵进行操作。

假设 ***A*** 是一个 $n \times n$ 的矩阵，***A*** 的特征值问题就是找到下面方程组的解：

$$A \cdot V = \lambda \cdot V$$

其中，λ 为标量，***V*** 为矢量，若把矩阵 ***A*** 的 n 个特征值放在矩阵 ***P*** 的对角线上，相应的

特征向量按照与特征值对应的顺序排列，作为矩阵 ***V*** 的列，特征值问题可以改写为：

$$A\cdot V=V\cdot D$$

如果 ***V*** 是非奇异的，该问题可以认为是一个特征值分解问题，此时关系式如下：

$$A=V\cdot D\cdot V^{-1}$$

广义特征值问题是指方程 ***A***·***x***=l ·***B***·***x*** 的非平凡解问题，其中 ***A***、***B*** 都是 *n*×*n* 的矩阵，l 为标量。满足此方程的l 为广义特征值，对应的向量 ***x*** 为广义特征向量。

如果 ***X*** 是一个列向量为 ***a*** 的特征向量的矩阵，并且它的秩为 *n*，那么特征向量线性无关。如果不是这样，则称矩阵为缺陷阵。如果 ***X'***·***X***=***I***，则特征向量正交，这对于对称矩阵是成立的。

2．特征值和特征向量的相关函数

现将 MATLAB 中矩阵特征值与特征向量的相关函数的具体调用格式及其功能列出：

- eig(A)：求包含矩阵 ***A*** 的特征值的向量。
- [X,D]=eig(A)：产生一个矩阵 ***A*** 的特征值在对角线上的对角矩阵 ***D*** 和矩阵 ***X***，它们的列是相应的特征向量，满足 ***AX***=***XD***。为了得到有更好条件特征值的矩阵，要进行相似变换。
- [T,B]=balance(A)：找到一个相似变换矩阵 ***T*** 和矩阵 ***B***，使得它们满足 ***B***=***T***-***A***·***T***。***B*** 是用命令 balance 求得的平衡矩阵。
- eig(A,'nobalance')：不经过平衡处理求得矩阵 ***A*** 的特征值和特征向量，也就是不进行平衡相似变换。
- eigs(A)：返回一个由矩阵 ***A*** 的部分特征值组成的向量，和 eig 命令一样，但是不返回全部的特征值。如果不带有参量，则计算出最大的特征值。当计算所有特征值时，如果矩阵 ***A*** 的秩不小于 6，则计算出 6 个特征值。
- eigs(f,n)：求出矩阵 ***A*** 的部分特征值。在使用一个矩阵列的线性运算符时，字符串 ***f*** 中包含的是 M 文件的文件名，***n*** 指定问题的阶次。用这种方法来求特征值比开始就用运算符来求要快。
- eigs(A,B,k,sigma)：求矩阵 ***A*** 的部分特征值，矩阵 ***B*** 的大小和 ***A*** 相同；如果没有给出 ***B***=eye(size(***A***))，那么 ***k*** 就是要计算的特征值的个数；如果 ***k*** 没有给出，就用小于 6 的数或者 ***A*** 的秩。

变量 sigma 是一个实数或者复数的移位参数，或者下列文本字符串中的一个，文本字符串指明的是特征值的属性：“lm”为最大的特征值，“sm”为最小的特征值，“lr”为最大的实数部分，“sr”为最小的实数部分，“be”为同时求得最大和最小的实数部分。

- condeig(A)：返回一个由矩阵 ***A*** 的特征值条件数组成的向量。
- [V,D,s]=condeig(A)：返回[***V***,***D***]=eig(***A***)和 s=condeig(***A***)。

3．特征值和特征向量的计算

例 3-50，矩阵特征值与特征向量的计算示例。

在命令行窗口中输入：

```
A = [0.8 0.2; 0.2 0.8];
[Q,d] = eig(A)
Q*Q'
```

输出结果：

```
Q =
   -0.7071    0.7071
    0.7071    0.7071
d =
    0.6000         0
         0    1.0000
ans =
    1.0000         0
         0    1.0000
```

3.5　稀疏矩阵

在许多问题中提到了含有大量 0 元素的矩阵，这样的矩阵称为稀疏矩阵。为了节省存储空间和计算时间，MATLAB 考虑到矩阵的稀疏性，在对它进行运算时有特殊的命令。

一个稀疏矩阵中有许多元素等于零，这便于矩阵的计算和保存。如果 MATLAB 把一个矩阵当作稀疏矩阵，那么只需在 m×3 的矩阵中存储 m 个非零项。第 1 列是行下标，第 2 列是列下标，第 3 列是非零元素值，不必保存 0 元素。如果存储每个浮点数需要 8 字节，存储每个下标需要 4 字节，那么整个矩阵在内存中存储需要 16×m 字节。

例 3-51，稀疏矩阵与普通矩阵示例。

在命令行窗口中输入：

```
A=eye(1000);     %得到一个 1000×1000 的单位矩阵
B=speye(1000);  %得到一个 1000×3 的矩阵，每行包含行下标、列下标及元素本身
```

上例中的矩阵 ***A*** 的存储需要 8MB 空间，而稀疏矩阵 ***B*** 只需 16KB 的空间，它所需空间只是单位矩阵的 0.2%。对于许多的广义矩阵也可这样来做。

前面章节中的算术和逻辑运算都适用于稀疏矩阵。而相对于普通矩阵来说，稀疏矩阵的计算速度更快，因为 MATLAB 只对非零元素进行操作，这是稀疏矩阵的第二个突出的优点。例如，上例中，2·***A*** 需要 100 万次浮点运算，而计算 2·***B*** 只需要 2000 次浮点运算。因为 MATLAB 不能自动创建稀疏矩阵，所以要用特殊的命令来得到稀疏矩阵。

稀疏矩阵大部分的元素是 0，因此只需存储非零元素的下标和元素值，这种特殊的存储方式可以节省大量的存储空间和不必要的运算。

3.5.1　稀疏矩阵的存储方式

对于稀疏矩阵，MATLAB 仅存储矩阵所有的非零元素的值及其位置（行号和列号）。显然，这对于具有大量 0 元素的稀疏矩阵来说是十分有效的。

设矩阵 $A=\begin{bmatrix}1 & 0 & 0 & 0\\0 & 5 & 0 & 0\\2 & 0 & 0 & 7\end{bmatrix}$是具有稀疏矩阵特征的矩阵，其完全存储方式是按列存储的全部 12 个元素：1,0,2,0,5,0,0,0,0,0,0,7。其稀疏存储方式为：(1,1),1,(3,1),2,(2,2),5,(3,4),7。

其中，括号内为元素的行列位置，后面为元素值。当矩阵非常“稀疏”时，会有效地节省存储空间。

3.5.2　稀疏矩阵的生成

MATLAB 提供了多种创建稀疏矩阵的方法。

- 利用 sparse 函数从满矩阵转换得到稀疏矩阵。
- 利用一些特定函数创建包括单位稀疏矩阵在内的特殊稀疏矩阵。

1．利用 sparse 函数建立一般的稀疏矩阵

稀疏矩阵指令的调用格式如表 3-4 所示。

表 3-4　稀疏矩阵的指令集

函数名称	表示意义
sparse(***A***)	由非零元素和下标建立稀疏矩阵 ***A***。如果 ***A*** 已是一个稀疏矩阵，则返回 ***A*** 本身
sparse(*m*,*n*)	生成一个 *m*×*n* 的所有元素都是 0 的稀疏矩阵
sparse(***u***,***v***,***a***)	生成大小为 max(***u***)×max(***v***)的稀疏矩阵。其中 ***u*** 和 ***v*** 是整数向量，***a*** 为实数或者复数向量
sparse(***u***,***v***,***a***,*m*,*n*)	生成一个 *m*×*n* 的稀疏矩阵，(u_i,v_i)对应值 a_i。向量 ***u***，***v*** 和 ***a*** 长度必须相同
spconvert(***D***)	生成一个稀疏矩阵。***D*** 共有 3 列，第 1 列为行下标，第 2 列为列下标，最后一列为元素值
full(***S***)	将稀疏矩阵 ***S*** 转换成一个满矩阵

例 3-52，输入一个稀疏矩阵示例。

在命令行窗口中输入命令：

```
S = sparse([1,2,3,4,5],[2,1,4,6,2],[10,3,-2,-5,1],10,12)
```

结果如下：

```
S =
   (2,1)       3
   (1,2)      10
   (5,2)       1
   (3,4)      -2
   (4,6)      -5
```

此外，sparse 函数还可以将一个满矩阵转换成一个稀疏矩阵，相应的调用格式如下：

S = sparse(X)：***X*** 为满矩阵。

例如矩阵 $A=\begin{bmatrix}1 & 0 & 0 & 0\\0 & 5 & 0 & 0\\2 & 0 & 0 & 7\end{bmatrix}$，

输入命令：

```
A = [1 0 0 0;0 5 0 0;2 0 0 7]
S = sparse(A)
```

得到：

```
S =
  (1,1)      1
  (3,1)      2
  (2,2)      5
  (3,4)      7
```

反之，MATLAB 提供了 full()函数把稀疏矩阵转换为满矩阵。full()函数的调用格式如下：

- A = full(S)：***S*** 为稀疏矩阵。

例如将上例中得到的稀疏矩阵 ***S*** 转换为满矩阵。

输入命令：

```
B = full(S)
```

得到：

```
B =
    1    0    0    0
    0    5    0    0
    2    0    0    7
```

例 3-53，将普通矩阵转换为稀疏矩阵示例。

在命令行窗口中输入如下语句：

```
clear all
A=rand(16,9)>0.95
S=sparse(A)  %创建稀疏矩阵
whos
```

输出结果：

```
A=   0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    1    0    0
    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0
    0    0    0    1    0    0    0    0    0
    0    0    0    0    0    0    0    0    0
    0    0    0    0    1    0    0    0    0
    0    0    0    0    0    0    0    0    0
    1    1    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0
    0    0    1    1    0    0    0    0    0
    0    0    0    0    0    0    0    0    0
    0    0    0    0    1    0    0    1    0
S=
  (9,1)      1
  (9,2)      1
 (14,3)      1
  (5,4)      1
 (14,4)      1
```

```
   (7,5)       1
  (16,5)       1
   (2,7)       1
  (16,8)       1
  Name      Size           Bytes  Class     Attributes
A        16x9             144  logical
     S        16x9              85  logical    sparse
```

例 3-54，查看稀疏矩阵中非零的信息示例。

在命令行窗口中输入如下语句：

```
clear all
a=[0 0 0 1;0 0 8 0;4 0 0 0;0 0 0 0];
S=sparse(a);           %创建稀疏矩阵
whos;
n1=nnz(S)              %查看非零元素的个数
n2=nonzeros(S)         %非零元素的值
n3=nzmax(S)            %查看稀疏矩阵的存储空间
spy(S)
n4=nnz(S)/prod(size(S))
```

输出结果：

```
  Name     Size           Bytes  Class     Attributes
  A        4x4            128  double
  S        4x4             56  double    sparse
n1 =    3
n2 =
    4
    8
    1
n3 =    3
n4 =   0.1875
```

spy()函数对稀疏矩阵中非零元素的分布进行图形化显示，如图 3-3 所示。采用 nnz(***S***)/prod(size(***S***))计算稀疏矩阵的非零元素密度。

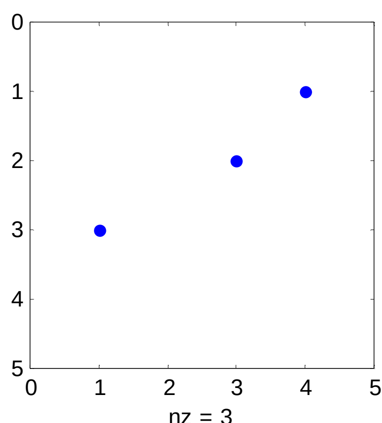


图 3-3　稀疏矩阵中非零元素的显示

2．利用特定函数建立稀疏矩阵

MATLAB 提供了一些函数来创建特殊的稀疏矩阵，这些函数如表 3-5 所示。

表 3-5　特殊稀疏矩阵的创建函数

函　　数	调用格式	描　　述
speye	S = speye(m,n)	创建单位稀疏矩阵
spones	S = spones(X)	创建非零元素为 1 的稀疏矩阵
sprand	S = sprand(X)	创建非零元素为均匀分布的随机数的稀疏矩阵
sprandn	S = sprandn(X)	创建非零元素为高斯分布的随机数的稀疏矩阵
sprandsym	S = sprandsym(X)	创建非零元素为高斯分布的随机数的对称稀疏矩阵
spdiags	S =spdiags(X)	创建对角稀疏矩阵
spalloc	S =spalloc(X)	为稀疏矩阵分配空间

例 3-55，利用 speye 函数创建单位稀疏矩阵示例。

在命令行窗口中输入如下语句：

```
clear all
A=speye(5)          %创建 5 阶单位稀疏矩阵
B=speye(5,6)        %创建 5×6 稀疏矩阵
C=full(A)
D=full(B)
```

输出结果：

```
A=
  (1,1)       1
  (2,2)       1
  (3,3)       1
  (4,4)       1
  (5,5)       1
B =
  (1,1)       1
  (2,2)       1
  (3,3)       1
  (4,4)       1
  (5,5)       1
C=
    1    0    0    0    0
    0    1    0    0    0
    0    0    1    0    0
    0    0    0    1    0
    0    0    0    0    1
D=
    1    0    0    0    0    0
    0    1    0    0    0    0
    0    0    1    0    0    0
    0    0    0    1    0    0
    0    0    0    0    1    0
```

例 3-56，创建非零元素为随机数的对称稀疏矩阵示例。

在命令行窗口中输入如下语句：

```
clear all
A=sprandsym(5,0.1)     %建立非零元素为随机数的对称稀疏矩阵
B=spones(a)            %建立非零元素为 1 的与矩阵 A 维数相同的对称稀疏矩阵
C=full(A)
D=full(B)
```

输出结果：

```
A=
   (5,1)     -0.0631
   (4,3)      0.7147
   (3,4)      0.7147
   (1,5)     -0.0631
B=
   (5,1)       1
   (4,3)       1
   (3,4)       1
   (1,5)       1
C=
         0         0         0         0   -0.0631
         0         0         0         0         0
         0         0         0    0.7147         0
         0         0    0.7147         0         0
   -0.0631         0         0         0         0
D=
     0     0     0     0     1
     0     0     0     0     0
     0     0     0     1     0
     0     0     1     0     0
     1     0     0     0     0
```

3.5.3　稀疏矩阵的运算

满矩阵的四则运算对稀疏矩阵同样有效，但是返回结果有可能是稀疏矩阵或者满矩阵。

对于单个稀疏矩阵的输入，大部分函数输出的结果都是稀疏矩阵，有部分函数输出的结果是满矩阵。对于多个矩阵的输入，如果其中至少有一个矩阵是满矩阵，那么大部分函数的输出结果是满矩阵。

对于矩阵的加、减、乘、除运算，只要其中有一个是满矩阵，则输出的结果都是满矩阵。

稀疏矩阵的数乘为稀疏矩阵；稀疏矩阵的幂为稀疏矩阵。

3.6　本章小结

本章介绍了 MATLAB 数组与矩阵的相关知识，主要内容包括数组运算、矩阵操作、矩阵元素运算、矩阵通用运算和稀疏矩阵等。本章是理解 MATLAB 运算方式的重点，因为在 MATLAB 中所有的数据都是通过数组或矩阵的方式组织并进行计算的。要掌握好本章的内容，还需要了解更多的知识，可查阅相关的书籍和 MATLAB 帮助文件等。

第 4 章

MATLAB 编程基础

知识要点

本章主要内容是对 MATLAB 平台上的基本编程进行讲解与分析。涉及的新概念有：M 文件编辑器、控制流、脚本、函数、局部变量、全局变量、子函数、私有函数、重载函数、eval 和 feval 函数、内联函数和 P 码文件等，还涉及 MATLAB 编程中的向量化、预分配、变量检测与传递等。针对每一部分都有相应的示例与讲解相配合，以使读者对这些抽象概念能够真正理解和掌握，并且体会到面向对象编程的优越性和重要性。

学习要求	知识点 \ 学习目标	了解	理解	应用	实践
	M 文件编辑器		√		
	变量			√	√
	MATLAB 控制流			√	√
	函数			√	√
	脚本			√	√
	变量检测和传递			√	√
	MATLAB 程序调试	√			√



4.1 M 文件编辑器

在前面几个章节的示例中，全部采用在 MATLAB 界面的命令行窗口的 MATLAB 提示符下输入指令并运行的方式。然而当需要完成的运算比较复杂，需要几十行甚至成百上千行指令来完成时，命令行窗口就不再适用了。

为了代替在命令行窗口中输入 MATLAB 指令的语句，MATLAB 平台上提供了一个文本文件编辑器，用来创建一个 M 文本文件来写入这些指令。M 文件的扩展名为.m。一个 M 文件包含许多连续的 MATLAB 指令，这些指令完成的操作可以是引用其他的 M 文件，也可以是引用自身文件，还可以进行循环和递归等。

（1）创建新的 M 文件，启动 M 文件编译器的操作方法如下：

①在 MATLAB 命令行窗口运行指令 edit。

②单击 MATLAB 主页中新建脚本的图标。

③选择 MATLAB 主页中新建图标下面的脚本。

（2）打开已有的 M 文件的操作方法如下：

①在 MATLAB 命令行窗口运行指令 edit filename。其中 filename 是已有的文件名，可以不带扩展名，文件名也可以省略不写。

②单击 MATLAB 主页中的按钮，再按照弹出对话框中的提示选择已有的 M 文件。

（3）经过修改的 M 文件的保存方法如下：

①单击 M 文件编辑器工具栏上的“保存”按钮。若是已有此 M 文件，则保存操作完成；若是新的 M 文件，则会弹出对话框，需要选择存放目录和文件名，才能完成 M 文件的保存。

②单击 MATLAB 菜单栏中的“保存”按钮，之后操作同①。

每当用户输入这个文件名的自变量时，MATLAB 就会执行文件中的所有指令。在这个过程中，MATLAB 从文件而不是从终端读取命令，当文件中最后一条命令被执行时，MATLAB 能再从终端读取命令。读取文件时，MATLAB 将首先在当前工作文件夹下寻找此文件，如果它不在当前文件夹下，则在该路径下的所有目录中搜索。该路径保存在 MATLAB path（参见表 4-1）中。

> **○ 注意**
>
> 如果想执行一个没有放在可以自动搜索处的文件，则可以单击 M 文件编辑器界面中的“运行”按钮。

在 MATLAB 的工具库中有大量的预定义 M 文件，例如 addpath.m、computer.m、copyfile.m 和 newplot.m 等，这些文件一般会在安装 MATLAB 软件时直接被存放在安装目录中，也可以使用命令 what，列出由用户定义的和在 MATLAB 目录中存放的 M 文件。

关于 M 文件编辑器和 M 文件，MATLAB 中提供了大量操作和控制指令，如表 4-1 与表 4-2 所示。

表 4-1　MATLAB 文件操作指令集

数据格式	说　　明
whatdirname	列出当前文件夹下所有的 MATLAB 文件。如果给定 dirname，就列出目录 dirname 下的文件
dirname	dir 列出一个目录或子目录中的所有文件。这个命令可以用不同的路径名和程序单
Dir	列出一个目录或子目录中的所有文件。这个命令可以用不同的路径名和程序单
ls	以不同的输出格式列出文件
Delete filename	删除文件 filename
cd	改变当前文件夹
Type filename	显示文件 filename 的内容。如果没有指定扩展名，则 MATLAB 就读 filename.m
Edit file	打开一个编辑器。如果给定 file，那么这个文件就在编辑器中打开
copyfile(file1, file2)	file1 复制到 file2。有关错误处理可参见 helpcopyfile
Which filename	显示由 filename 指定的函数的搜索路径
path	显示 MATLAB 的目录搜索路径。如果给出带自变量的命令，就改变搜索路径。输入 helppath 可以获得更多信息
MATLABpath	当一个新的搜索路径给定时，作为工作路径，但没有错误处理
genpath(directory)	返回一个新的搜索路径，这个路径是由老的和在 MATLABROOT/toolbox 下所有路径一起组成的。如果 directory 给定，那么在 directory 下所有的目录都被增加替换
pathsep	列出分隔标志
partialpath	列出本地搜索路径
Editpath	给出一个图形用户界面。在那里，可以从 MATLAB 的搜索路径增加和编辑目录
addpath(dir1,dir2,…,flag)	在 MATLAB 的搜索路径的开头增加目录。如果字符串 flag 给定，且是始端，那么目录被加在始端；如果它是末端，则加在末端
rmpathdir	从 MATLAB 的搜索路径中移去目录 dir
pathtool	这是一个修改搜索路径的图形工具。尽管 helppathtool 建议它不要在 UNIX 下工作，但它还是在 UNIX 下工作
path2rc	在文件 pathdef.m 中保存当前的搜索路径，当启动 MATLAB 时，可以从这个文件中读取搜索路径
dbtypefilename	带行号显示文件 filename 的内容。如果在 filename 中没有给定扩展名，则 MATLAB 使用扩展名.m
lasterr	重复上次的错误信息
lastwarn	重复上次的警告信息
isdirdirname	如果 dirname 是一个目录，则返回 1；否则返回 0
dos	从 MATLAB 中运行一个 DOS 命令
vms	从 MATLAB 中运行一个 VMSDCL 命令
unix	从 MATLAB 中执行一个 UNIX 操作系统命令
tempdir	返回一个表示系统中临时目录名的字符串
tempname	返回一个以“tp”开头的字符串，MATLAB 将检查这个字符串是否为系统的临时目录中的一个文件名。这个字符串可以用作一个临时文件的名字
MATLABroot	返回带指向 MATLAB 安装所在目录的搜索路径的一个字符串

表 4-2　MATLAB 文件流控制指令集

数据格式	说　明
runfilename	运行命令文件 filename。filename 包括文件的全部路径和文件名
pause	暂停 M 文件的运行，按任意键后继续运行
pause(n)	暂停运行 *n* 秒后继续执行。这个暂停命令在显示大量图形时非常有用
pauseoff	指示 MATLAB 跳过后面的暂停
pauseon	指示 MATLAB 遇到暂停时执行暂停命令
break	终止 for 和 while 循环。如果在一个嵌套循环中使用该命令，则只有内部循环被终止
return	结束 M 文件运行，MATLAB 立即返回到函数被调用的地方
error(str)	终止 M 文件的运行，并在屏幕上显示错误信息和字符串 str
errortrapstate	决定当有错误发生时是否停止运行。state 的值可为 on（此时捕获错误信息并继续执行）或 off（发生错误时停止运行）
global	声明变量为全局变量。全局变量能在函数文件中被访问，而不必包括在参数列表中。命令 global 后面是以空格分开的变量列表。声明为全局变量将保持其全局性直至工作区完全被清除掉或使用了 clearglobal 命令
isglobal(name)	如果变量 name 是全局变量，则返回 1；否则返回 0
mfilename	返回正在运行的 M 文件名字符串，一个函数能用这个函数获得它自己的名字

4.2　变量

在程序中会经常定义一些变量来保存和处理数据。从本质上看，变量代表了一段可操作的内存，也可以认为变量是内存的符号化表示。

当程序中需要使用内存时，可以定义某种类型的变量。此时编译器根据变量的数据类型分配一定大小的内存空间，程序就可以通过变量名来访问对应的内存。本节将介绍 MATLAB 中变量的相关知识。

4.2.1　变量的命名

在 MATLAB 中，变量不需要预先声明就可以进行赋值。变量的命名遵循以下规则：

- 变量名和函数名对字母的大小写敏感，因此 *x* 和 *X* 是两个不同的变量；sin 是 MATLAB 定义的正弦函数，而 SIN 不是。
- 变量名必须以字母开头，其后可以是任意字母或下画线，但是不能有空格、中文或标点，例如_xy、a.b 均为不合法的变量名，而 classNum_x 是一个合法的变量名。
- 不能使用 MATLAB 的关键字作为变量名。避免使用函数名作为变量名。如果变量采用函数名，则该函数失效，例如设置变量名为“if”“end”等。
- 变量名最多可包含 63 个字符，从第 64 个字符开始之后的字符将被忽略。为了程序可读性及维护方便，变量名一般代表一定的含义。

通过调用 isvarname 函数，可以验证用户指定的变量名是否为能被 MATLAB 接收的合

法变量名。该函数返回值为 1 或者 0，表示合法或者不合法。例如：

```
isvarname('_xy')
ans =    0           %返回值 0 表明该变量名不合法
isvarname('classNum_x')
ans =    1           %返回值 1 表明该变量名合法
```

4.2.2　变量的类型

MATLAB 将变量划分为三类：局部变量、全局变量和永久变量。

（1）局部变量。MATLAB 中的每一个函数都有自己的局部变量，这些变量存储在该函数独立的工作区中，与其他函数的变量及主工作区中的变量分开存储。当该函数调用结束后，这些变量随之被删除，不会保存在内存中。

（2）全局变量。全局变量在定义该变量的全部工作区中有效。当在一个工作区内改变该变量的值时，该变量在其余工作区内的值也将改变。

通常，全局变量的变量名用大写字母来表示，并在函数体的开头位置进行定义，其格式如下：

```
global X_Val
```

使用全局变量的目的是减少数据传递的次数。然而，使用全局变量有一定的风险，容易造成错误，这种错误源自全局变量的工作原理。

（3）永久变量。永久变量用 persistent 声明，只能在 M 文件函数中定义和使用，只允许声明它的函数存取。当声明它的函数退出时，MATLAB 不会从内存中清除它。例如声明 *a* 为永久变量：

```
persistent a
```

4.2.3　MATLAB 默认的特殊变量

MATLAB 预定义了许多特殊变量，这些变量具有系统默认的含义，如表 4-3 所示。

表 4-3　MATLAB 中的特殊变量

特殊变量	描　　述
ans	系统默认的用作保存运算结果的变量名
pi	圆周率
eps	机器零阈值，MATLAB 中的最小值
inf	表示无穷大
NaN 或 nan	表示不定数
i 或 j	虚数
nargin	函数的输入参数个数
nargout	函数的输出参数个数

（续表）

特殊变量	描　述
realmin	可用的最小正实数
realmax	可用的最大正实数
bitmax	可用的最大正整数（以双精度格式存储）
varargin	可变的函数输入参数个数
varargout	可变的函数输出参数个数
beep	使计算机发出“嘟嘟”声音

例 4-1，特殊变量的应用示例。

根据圆的面积计算公式 $S=\pi r^2$，计算半径为 6 的圆的面积。输入命令：

```
pi*(6^2)
```

得到结果：

```
ans =  113.0973
```

4.2.4　关键字

关键字是 MATLAB 程序设计中常用到的流程控制变量，共有 20 个。在命令行窗口中输入命令 iskeyword，即可查询这 20 个关键字：

```
iskeyword'
ans =     'break'   'case'    'catch'    'classdef'    'continue'    'else'
        'elseif'    'end'    'for'    'function'    'global'    'if'
    'otherwise'  'parfor'   'persistent'    'return'    'spmd'    'switch'
    'try'
        'while'
```

如果用户将这些关键字作为变量名，则 MATLAB 会出现错误提示。

4.3　MATLAB 的控制流

MATLAB 平台上的控制流结构包括顺序结构、if-else-end 分支结构、switch-case 结构、try-catch 结构、for 循环结构和 while 循环结构，这 6 种结构的算法及使用与其他计算机编程语言十分类似，有编程基础的读者可以很快掌握。

4.3.1　顺序结构

顺序结构是 MATLAB 程序中最基本的结构，表示程序中的各操作是按照它们出现的先后顺序执行的。顺序结构可以独立使用构成一个简单的完整程序，常见的输入、计算、输出三部曲的程序就是顺序结构。在大多数情况下，顺序结构作为程序的一部分，与其他结构一起构成一个复杂的程序，例如分支结构中的复合语句、循环结构中的循环体等。

例 4-2，计算圆的面积。该程序的语句顺序就是输入圆的半径 r，计算 $S=\pi\cdot r\cdot r$，输出圆

的面积 *S*。试着在 MATLAB 中编写求解圆面积的顺序结构的程序。

调出 MATLAB 的程序编辑器。单击编辑器工具栏中的“新建”图标，系统即新建一个默认名为 Untitled 的 M 文件，如图 4-1 所示。

在输入区输入下述代码（如图 4-2 所示）：

```
% 定义变量r，并赋值
r = 5;
% 计算圆的面积
S = pi*r*r;
% 输出面积
fprintf('Area = %f\n',S);
```

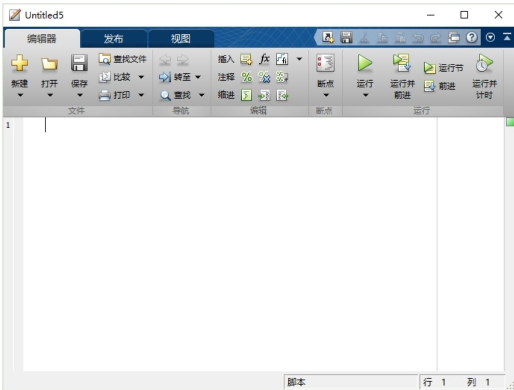


图 4-1　新建 M 文件

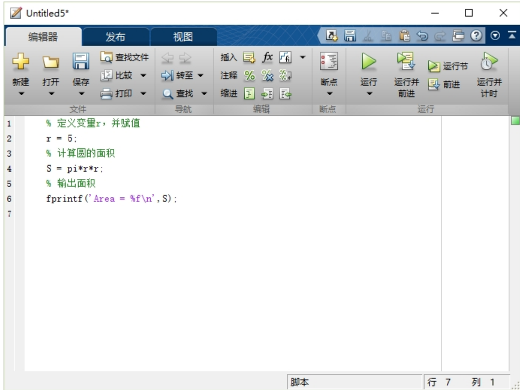


图 4-2　输入程序

单击编辑器工具栏中的“保存”图标，并以“ex4_2”为名称保存。单击工具栏上的“运行”图标运行该 M 文件，得到结果：

```
Area = 78.539816
```

4.3.2　if-else-end 分支结构

if-else-end 指令为程序流提供了一种分支结构，该结构的形式根据实际情况的不同而不同，主要有以下几种：

（1）若判决条件 expression 为真，则执行命令组，否则跳过该命令组。具体的调用格式如下：

```
if expression
        commands
end
```

（2）若可供选择的执行命令组有两组，则采用如下结构：

```
if expression               %判决条件
        commands1           %判决条件为真，执行命令组 1，并结束此结构
else
        commands2           %判决条件为真，执行命令组 2，并结束此结构
end
```

（3）若可供选择的执行命令组有 *n*（*n*>2）组，则采用如下结构：

```
if expression1          %判决条件
```

```
        commands1    %判决条件 expression1 为真，执行 commands1，并结束此结构
elseif expression
        commands2    %判决条件 expression1 为假，expression2 为真，执行 commands2，
        并结束此结构
…
else
        commandsn    %前面所有判决条件均为假，执行 commandsn，并结束此结构
end
```

例 4-3，if-else-end 分支结构的简单运用示例。

创建 M 文件并命名为 ex4_3.m。

利用 M 文件编辑器，在 ex4_3.m 文件中写入：

```
Rand_a = rand(1) %创建一个随机数 Rand_a
if Rand_a > 0.5
Rand_b = Rand_a   %如果 Rand_a 大于 0.5，那么创建变量 Rand_b 并等于 Rand_a
else
Rand_b=1-Rand_a %如果 Rand_a 不大于 0.5，那么创建变量 Rand_b 并等于 1-Rand_a
end
```

运行 M 文件，得到结果：

```
Rand_a = 0.3678
Rand_b = 0.6322   %由于产生的随机数 Rand_a 小于 0.5，因此 Rand_b 等于 1-Rand_a
```

例 4-4，已知符号函数

$$y = \operatorname{sgn} x = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$$

使用 if 语句判断当给定变量 x 的值时，相应的函数值 y。

打开 Editor（M 文件编辑器），编写程序如下：

```
x = input('enter''x'':');
if(x>0)
   y = 1;
elseif(x==0)
   y = 0;
else
   y =-1;
end
disp(y)
```

将该脚本 M 文件命名为 ex4_4.m。调用 ex4_4.m，分别输入不同的 x 值，运行程序，得到结果：

```
ex4_4
enter'x':10
    1
```

4.3.3　switch-case 结构

switch 语句执行基于变量或表达式值的语句组，关键字 case 和 otherwise 用于描述语句组。只执行第一个匹配的情形。用到 switch 则必须用 end 与之搭配。switch-case 的具体语法

结构如下：

```
switch value     %value 为需要进行判决的标量或字符串
    case test1
        command1 %如果 value 等于 test1，执行 command1，并结束此结构
    case test2
        command2 %如果 value 等于 test2，执行 command2，并结束此结构
…
    case testk
        commandk %如果 value 等于 testk，执行 commandk，并结束此结构
otherwise
        commands %如果 value 不等于前面所有值，执行 commands，并结束此结构
end
```

说明：

（1）switch-case 结构的调用格式保证了至少有一组指令组将会被执行。

（2）switch 指令之后的表达式 value 应为一个标量或一个字符串。当表达式为标量时，比较命令为表达式==检测值 i；而当表达式为字符串时，MATLAB 将会调用字符串函数 strcmp 来进行比较，strcmp(表达式,检测值 i)。

（3）case 指令之后的检测值不仅可以是一个标量或一个字符串，还可以是一个元胞数组。如果检测时是一个元胞数组，则 MATLAB 将会把表达式的值与元胞数组中的所有元素进行比较。如果元胞数组中有某个元素与表达式的值相等，MATLAB 则认为此次比较的结果为真，从而执行与该次检测相对应的命令组。

例 4-5，switch-case 结构的简单运用示例。

创建 M 文件并命名为 ex4_5.m。

利用 M 文件编辑器，在 ex4_5.m 文件中写入：

```
num = 3;
switch num
   case 1
      data = 'Monday'     %如果 num=1，定义 data='Monday'
   case 2
      data = 'Tuesday'        %如果 num=2，定义 data='Tuesday'
   case 3
      data = 'Wednesday'  %如果 num=3，定义 data='Wednesday'
   case 4
      data = 'Thursday'       %如果 num=4，定义 data='Thursday'
   case 5
      data = 'Friday'     %如果 num=5，定义 data='Friday'
   case 6
      data = 'Saturday'       %如果 num=6，定义 data='Saturday'
   case 7
      data = 'Sunday'     %如果 num=7，定义 data='Sunday'
otherwise
      data = 'None!!!'        %如果 num 不等于上面所有值，定义 data = 'None!!!'
end
```

运行 M 文件，得到结果：

```
data =Wednesday
```

例 4-6，求任意底数的对数函数值 $y = \log_n x$。

打开 Editor（M 文件编辑器），编写程序如下：

```
clear
n = input('Enter the value of''n'':');
x = input('Enter the value of''x'':');
switch(n)
    case 1
    errordlg('出错');
    case 2
    y=log2(x);
    case exp(1)
    y=log(x);
    case 10
    y=log10(x);
   otherwise
    y=log10(x)/log10(n);
end
disp(y)
```

将该脚本 M 文件命名为 ex4_6.m。调用 ex4_6.m，分别输入不同的 *n* 值和 *x* 值，运行程序，输出结果：

```
ex4_6
Enter the value of'n':2
Enter the value of'x':10
   3.3219
ex4_6
Enter the value of'n':6
Enter the value of'x':32
   1.9343
```

当 n=1 时，输入 x 值后，出现错误提示对话框，如图 4-3 所示。

```
ex4_6
Enter the value of'n':1
Enter the value of'x':5
未定义函数或变量 'y'。
```

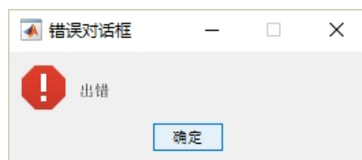


图 4-3　错误提示对话框

注意

与多分支的 if 语句相比，switch 语句主要用于条件多且单一的情况，典型的应用情况是数学中的分段函数。此外，两者各有优缺点，如表 4-4 所示。

表 4-4　if 语句和 switch 语句的比较

if 语句	switch 语句
比较复杂，特别是嵌套使用的 if 语句	可读性强，容易理解
要调用 strcmp 函数比较不同长度的字符串	可比较不同长度的字符串
可检测相等和不相等	仅检测相等

4.3.4　try-catch 结构

try-catch 结构的具体句法形式如下：

```
try
    command1 %命令组 1 总是首先被执行。若正确，执行完成后结束此结构
catch
    command2 %命令组 1 执行发生错误时，执行命令组 2
end
```

说明：

（1）只有当 MATLAB 执行命令组 1 发生错误时，才执行命令组 2。try-catch 结构只提供两个可供选择的命令组。

（2）当执行 command1 发生错误时，可调用 lasterr 函数查询出错的原因。如果函数 lasterr 的运行结果为空字符串，则表示命令组 1 被成功执行了。

（3）如果执行命令组 2 时又发生错误，则 MATLAB 将会终止该结构。

例 4-7，ry-catch 结构的简单运用示例。

创建 M 文件并命名为 ex4_7.m。

利用 M 文件编辑器，在 ex4_7.m 文件中写入：

```
Num = 6;
Mat = magic(4)                 %生成一个 4×4 的矩阵 Mat
try
   Mat_Num = Mat(Num,:)        %取 Mat 的第 Num 行元素
catch
   Mat_end = Mat(end,:)        %若 Mat 没有第 Num 行元素，则取 Mat 的最后一行元素
end
lasterr                        %显示出错原因
```

运行 M 文件，得到结果：

```
Mat =
   16    2    3   13
    5   11   10    8
    9    7    6   12
    4   14   15    1
Mat_end =
    4   14   15    1
ans =
Attempted to access Mat(6,:); index out of bounds because size(Mat)=[4,4].
```

4.3.5　for 循环结构

for 循环结构是针对大型运算相当有效的运算方法。MATLAB 中提供的循环结构有 for 循环结构和 while 循环结构两种。for 循环重复执行一组语句一个预先给定的次数，匹配的 end 描述该语句。for 循环的具体句法形式如下：

```
for x=array
    commands
end
```

说明：

（1）for 指令后面的变量 x 称为循环变量，而 for 与 end 之间的组命令 commands 被称为循环体。循环体被重复执行的次数是确定的，该次数由 array 数组的列数来确定。因此，在 for 循环过程中，循环变量 x 被依次赋值为数组 array 的各列，每次赋值，循环体都被执行一次。

（2）for 循环内部语句末尾的分号隐藏重复的打印，若 commands 指令中包含变量，则循环后在命令行窗口直接输入变量 r 来显示变量 r 经过循环后的最终结果。

例 4-8，利用 for 循环创建对称矩形。

创建 M 文件并命名为 ex4_8.m。

利用 M 文件编辑器，在 ex4_8.m 文件中写入：

```
for i = 1:4
   for j = 1:4
      if i>(5-j)
      else
         Mat(i,j) = i + j - 1;
      end
   end
end
Mat
```

运行 M 文件，得到结果：

```
Mat =
    1    2    3    4
    2    3    4    0
    3    4    0    0
    4    0    0    0
```

例 4-9，利用 for 循环求解 1+2+⋯+100 的和，即 $\sum_{i=1}^{100} i$。

在 M 文件 for_loop 中输入如下语句：

```
sum = 0;
for i = 1:1:100
   sum = sum + i;
end
sum
```

运行后可得到如下结果：

```
sum =
      5050
```

例 4-10，利用 for 循环嵌套求解 $x=\sin(\frac{n\cdot k\cdot\pi}{360})$，$n\in[1:10]$，$k\in[1:4]$。

在命令行窗口中输入如下语句：

```
x = [];
for n = 1:1:10
   for k = 1:1:4
      x(n,k) = sin((n*k*pi)/360);
   end
end
```

```
x
```

运行后可得到如下结果：

```
x =
   0.0087   0.0175   0.0262   0.0349
   0.0175   0.0349   0.0523   0.0698
   0.0262   0.0523   0.0785   0.1045
   0.0349   0.0698   0.1045   0.1392
   0.0436   0.0872   0.1305   0.1736
   0.0523   0.1045   0.1564   0.2079
   0.0610   0.1219   0.1822   0.2419
   0.0698   0.1392   0.2079   0.2756
   0.0785   0.1564   0.2334   0.3090
   0.0872   0.1736   0.2588   0.3420
```

4.3.6　while 循环结构

while 循环在一个逻辑条件的控制下重复执行一组语句一个不定的次数，匹配的 end 描述该语句。while 循环的具体句法形式如下：

```
while expression
    commands
end
```

说明：

（1）在 while 和 end 之间的命令组被称为循环体。MATLAB 在运行 while 循环之前，首先检测 expression 的值，若其逻辑值为真，则执行命令组；命令组第一次执行完毕后，继续检测 expression 的逻辑值，若其逻辑值仍为真，则循环执行命令组，直到表达式 expression 的逻辑值为假时，结束 while 循环。

（2）while 循环和 for 循环的区别在于，while 循环结构的循环体被执行的次数是不确定的，而 for 循环中循环体的被执行次数是确定的。

（3）一般情况下，表达式的值都是标量值，但是 MATLAB 中也同样运行表达式为数组的情况。若表达式为数组且数组所有元素的逻辑值均为真时，while 循环才继续执行命令组。

（4）如果 while 指令后的表达式为空数组，那么 MATLAB 默认表达式的值为假，直接结束循环。

（5）在 if-else-end 分支结构中提到的有关变量比较的注意事项，对 while 语句也同样适用。

例 4-11，while 结构的简单运用示例。

Fibonacci 数列的元素满足如下规则：$a_{k+2}=a_k+a_{k+1}$（k=1,2,⋯）；而且 $a_1=a_2=1$。现在要求出 Fibonacci 数列中第一个大于 9999 的元素。

利用 M 文件编辑器，在 ex4_11.m 文件中写入：

```
a(1) = 1;
a(2) = 1;
i = 2;
while a(i) < 10000
   a(i+1) = a(i) + a(i-1);        %当元素小于等于 9999 时，求下一项
   i = i + 1;
end
```

```
[i a(i)]
```

运行 M 文件，得到结果：

```
ans =21        10946         %Fibonacci 数列中第 21 项 10946 是第一个大于 9999 的数
```

例 4-12，利用 while 循环求解表达式 $\sum_{i=1}^{100} i$。

在 M 文件中输入如下语句：

```
i = 1;
sum = 0;
while i<101
    sum = sum + i;
    i = i + 1;
end
sum
```

运行程序，得到结果：

```
sum =   5050
```

for 循环和 while 循环的对比：

for 循环与 while 循环的区别在于 for 循环的循环次数是一定的，由(end-start)/initval 决定；而 while 循环的循环次数是不确定的。

因此，当用户无法确定循环次数，或者根本不需要知道循环次数，而只需要确定满足什么条件循环不停止的情况下，使用 while 循环比较合理。

4.4 控制程序流的其他常用指令

4.4.1 return 指令

通常，当被调用函数执行完成后，MATLAB 会自动将控制权转回主函数或 Commands 窗口。但是如果在被调用函数中插入 return 指令，可以强制 MATLAB 结束执行该函数并把控制权转出。

4.4.2 input 和 keyboard 指令

（1）input 指令将 MATLAB 的“控制权”暂时交给用户，用户通过键盘输入数值、字符串或表达式等，并按 Enter 键将输入内容传递到工作区，同时把“控制权”交还给 MATLAB。其常用的调用格式如下：

- Value = input('message')：将用户输入的内容赋值给变量 Value。
- Value = input('message', 's')：将用户输入的内容以字符串的形式赋值给变量 Value。

说明：

① 指令中的“message”是显示在屏幕上的字符串。

② 对于上面第一种调用格式，用户可以输入数值、字符串等各种形式的数据。

③ 对于上面第二种调用格式，用户无论输入什么内容，均以字符串的形式赋值给变量。

（2）当执行遇到 keyboard 指令时，MATLAB 将“控制权”暂时交给键盘，用户可以由键盘输入各种合法的 MATLAB 指令。只有当用户输入完成，并输入 return 指令后，“控制权”才交还给 MATLAB。

input 指令和 keyboard 指令的不同之处在于：keyboard 指令允许输入任意多个 MATLAB 指令，而 input 指令只允许用户输入赋值给变量的“值”，即数组、字符串或元胞数组等。

4.4.3　yesinput 指令

yesinput 指令是一个只能输入的指令，它提供的输入值是一个默认量，并可以对输入范围进行检查，其调用格式如下：

- Value = yesinput('Prompt', Default, Possib)。

说明：

（1）yesinput 指令涉及用户和 MATLAB 之间的交互，因此无法在 notebook 程序中运行。

（2）Prompt 为文字提示，Default 为默认的设置值，Possib 为可选值。

（3）当 yesinput 指令运行后，如果用户不输入任何值，则变量 Value 将接收默认值。

4.4.4　pause 指令

pause 指令的功能为控制执行文件的暂停与恢复，其调用格式如下：

- pause：暂停执行文件，等待用户按任意键继续。
- pause(n)：在继续执行文件之前，暂停 *n* 秒。

4.4.5　continue 指令

continue 语句把控制传给下一个在其中出现的 if 或 while 循环的迭代，忽略任何循环体中保留的语句。在嵌套循环中，continue 把控制传给下一个 for 或 while 循环所嵌套的迭代。

例 4-13，continue 指令使用示例。

本例展示了一个在 magic.m 文件代码中计算行数的 continue 循环，跳过所有空行和注释。continue 语句用于前进到 magic.m 的下一行。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
fid = fopen('magic.m','r');
count = 0;
while ~feof(fid)
   line = fgetl(fid);
   if isempty(line) | strncmp(line,'%',1)
      continue
   end
   count = count + 1;
end
disp(sprintf('%d lines',count));
```

在命令行窗口中输入上面的语句，得到结果：

```
31 lines
```

4.4.6 break 指令

在 for 循环或 while 循环结构中，有时并不需要运行到最后一次循环用户就已经得到了所需要的结果，那么后面的循环就变成冗余的了，消耗了运算时间并占用了内存。break 指令可进行对 for 循环或 while 循环结构的终止，通过使用 break 指令，可以不必等待循环的预定结束时刻，而是根据循环内部设置的终止项来判断。若终止项满足，则可以使用 break 指令退出循环；若终止项始终未满足，则照常运行至循环的预定结束时刻。

例 4-14，for 循环的中途终止。

利用 M 文件编辑器，在 ex4_14.m 文件中写入：

```
a(1) = 1;
a(2) = 1;
n = 1000;
for i = 3:n
   a(i) = a(i-1) + a(i-2);          %求下一项
   if a(i) > 9999
      [i a(i)]
      Break                         %当元素大于 9999 时，退出循环
   end
end
```

运行 M 文件，得到结果：

```
ans =     21     10946    %Fibonacci 数列中第 21 项 10946 是第一个大于 9999 的数
```

4.4.7 error 和 warning 指令

在编写 M 文件时，常用的错误或警告指令的调用格式有以下几种：

- error('message')：显示出错信息 message，终止程序。
- errortrap：错误发生后，控制程序继续执行与否的开关。
- lasterr：显示 MATLAB 系统判断的最新出错原因，并终止程序。
- warning('message')：显示警告信息 message，继续运行程序。
- lastwarn：显示 MATLAB 系统给出的最新警告程序，并继续运行。

4.5 脚本和函数

4.5.1 脚本

对于比较简单的运算过程，从命令行窗口中直接输入指令并运行计算是非常方便的。随着指令行的增加，或是运算逻辑复杂度的增加，以及重复计算要求的提出，再直接从命令行

窗口进行运算就十分不明智了。

在这种情况下，使用脚本文件最为适宜。“脚本”本身反映这样一个事实：MATLAB 是按照文件中所输入的指令执行的，这种文件的构成比较简单，其主要特点如下：

- 文件只是一串按照用户意愿排列而成的 MATLAB 指令集合。
- 脚本文件运行后，其运算过程中所产生的所有变量都自动保留在 MATLAB 工作区（Base 工作区）中，除非用户关闭 MATLAB 运行界面，或是使用 clear 指令对工作区中的变量加以清理，否则这些变量将一直保存在基本的工作区中。基本空间随着 MATLAB 的启动而产生，只有关闭 MATLAB 界面时，该基本空间才会被删除。
- 当调用一个脚本时，MATLAB 会简单地执行文件中找到的命令。脚本可以运行工作区中存在的数据，或者创建新数据来运行。
- 虽然脚本不能返回输出变量，但是所有创建的变量都将保留在工作区中，供后面的计算使用。另外，脚本能提供图形输出，就像使用图形输出函数 plot()一样。

例 4-15，脚本文件的简单运用示例。

利用 M 文件编辑器，在 ex4_15.m 文件中写入：

```
array = zeros(1,32);
for n = 3:32                    %for 循环的运用
array (n) = rank(magic(n));
end
array
bar(array)                      %用柱状图输出结果
```

运行 M 文件，结果如图 4-4 所示。

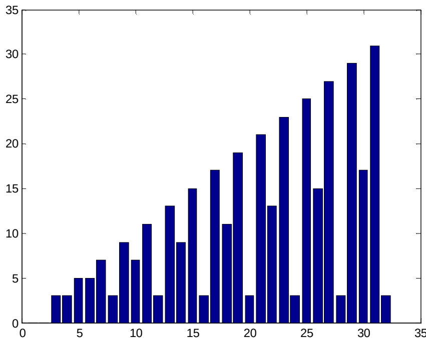


图 4-4　脚本文件运行结果

4.5.2　函数

如果 M 文件的第一个可执行语句以 function 开始，该文件就是函数文件，每一个函数文件都定义一个函数。事实上，MATLAB 提供的函数命令大部分都是由函数文件定义的，这足以说明函数文件的重要性。

从使用的角度看，函数是一个“黑箱”，把一些数据送进去，经加工处理，把结果送出来。从形式上看，函数文件区别于脚本文件之处在于脚本文件的变量为命令工作空间变量，

在文件执行完成后保留在命令工作空间中；而函数文件内定义的变量为局部变量，只在函数文件内部起作用，当函数文件执行完后，这些内部变量将被清除。

例 4-16，编写函数 average( )用于计算向量元素的平均值。

在 MATLAB 编辑窗口输入：

```
function y=average(x)
[a,b]=size(x);                          % 判断输入量的大小
if~((a==1)|(b==1))| ((a==1)& (b==1))    % 判断输入是否为向量
   error('必须输入向量。')
end
y=sum(x)/length(x);                     %计算向量 x 所有元素的平均值
```

将文件存盘，默认状态下函数名为 average.m(文件名与函数名相同)，函数 average 接收一个输入参数并返回一个输出参数，该函数的用法与其他 MATLAB 函数一样；在 MATLAB 命令行窗口中运行以下语句，便可求得 1～9 的平均值。

```
>> x=1:9
x =
    1    2    3    4    5    6    7    8    9
>> average(x)
ans =
    5
```

通常函数文件由以下几个基本部分组成。

（1）函数定义行：函数定义行由关键字 function 引导，指明这是一个函数文件，并定义函数名、输入参数和输出参数，函数定义行必须为文件的第一个可执行语句，函数名与文件名相同，可以是 MATLAB 中任何合法的字符。

函数文件可以带有多个输入和输出参数，如：

```
function [x,y,z]=sphere(theta,phi,rho)
```

也可以没有输出参数，如：

```
function printresults(x)
```

（2）H1 行：H1 行就是帮助文本的第一行，是函数定义行下的第一个注释行，是供 lookfor 查询时使用的。一般来说为了充分利用 MATLAB 的搜索功能，在编制 M 文件时，应在 H1 行中尽可能多地包含该函数的特征信息。由于在搜索路径上包含 average 的函数很多，因此用 lookfor average 语句可能会查询到多个有关的命令。如：

```
>> lookfor average_2
```

函数 average_2(*x*)用以计算向量元素的平均值。

（3）帮助文本：在函数定义行后面，连续的注释行不仅可以起到解释与提示作用，更重要的是为用户自己的函数文件建立在线查询信息，以供 help 命令在线查询时使用。如：

```
>> help average_2
```

函数 average_2(*x*)用以计算向量元素的平均值。

输入参数 *x* 为输入向量，输出参数 *y* 为计算的平均值。非向量输入将导致错误。

（4）函数体：函数体包含了全部的用于完成计算及给输出参数赋值等工作的语句，这些语句可以是调用函数、流程控制、交互式输入/输出、计算、赋值、注释和空行。

（5）注释：以%起始到行尾结束的部分为注释部分，MATLAB 的注释可以放置在程序的任何位置，可以单独占一行，也可以在一个语句之后。如：

```
%非向量输入将导致错误
[m,n]=size(x);   %判断输入量的大小
```

4.5.3　M 文件的一般结构

从结构上来看，脚本文件只是比函数文件少了一个“函数声明行”，除此之外，二者的语法及构架等均相同。于是将典型规范的 M 文件函数的结构总结如下：

- 函数声明行：位于函数文件的首行，以 MATLAB 关键字 function 开头，定义函数名及函数的输入/输出变量。脚本文件无须函数声明行。
- H1 行：紧随函数声明行之后的以“%”开头的第一注释行。H1 行包括大写的函数文件名和运用关键词简要描述的函数功能。该行将提供 lookfor 命令作为关键词查询和 help 在线帮助的使用。
- 在线帮助文本区：H1 行及其后的连续以“%”开头的注释行，通常包括函数输入/输出变量的含义调用说明。
- 编写和修改记录：与在线帮助文本区应以一个“空”行相隔；该行以“%”开头，记录了编写及修改 M 文件的所有的作者、日期及版本号，以方便后来的使用者查询、修改和使用。
- 函数主体：规范化的写法应与编写和修改记录以一个“空”行相隔；这部分内容包括所有实现该 M 函数文件功能的 MATLAB 指令、接收输入变量、进行程序流控制。

说明：

（1）函数定义名应和文件保存名一致。当两者不一致时，MATLAB 将忽视文件首行的函数定义名，而以文件保存名为准。

（2）MATLAB 中的函数文件名必须以字母开头，可以是字母、下画线及数字的任意组合，但不可以超过 31 个字符。

（3）建议读者在编写 H1 行注释时，尽量采用英文表述方式，这是为了之后的使用过程中关键词检索的方便。

例 4-17，完整的 M 文件示范。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
function spir_len = spirallength(d, n, lcolor)
%CIRCLE plot a circle of radius as r in the provided color and calculate its area
% d：螺旋的旋距
%   n：螺旋的圈数
%   lcolor：画图线的颜色
%   spir_len：螺旋的周长
% spirallength(d, n)：利用蓝色以预设参数的螺旋线
% spirallength(d, n,lcolor)：利用 lcolor 颜色以预设参数的螺旋线
% spir_len = spirallength(d, n)：计算螺旋线的周长，并用蓝色填充螺旋线
% spir_len = spirallength(d, n,lcolor)：计算螺旋线的周长，并用 lcolor 颜色填充螺旋线
% 编写于 2009.7.6，修改于 2011.8.8                程序员：01
```

```
if nargin > 3
   error('输入变量过多! ');
elseif nargin == 2
   lcolor = 'b';
end
j = sqrt(-1);
phi = 0: pi/1000 : n*2*pi;
amp = 0: d/2000 : n*d;
spir = amp .* exp(j*phi);
if nargout == 1
   spir_len = sum(abs(diff(spir)));
   fill(real(spir), img(spir), lcolor)
elseif nargout == 0
   plot(spir,lcolor)
else
   error('输出变量过多! ');
end
axis('square')
```

在命令行窗口中输入：

```
spirallength(0.25, 4, 'k:')
```

输出结果如图 4-5 所示。

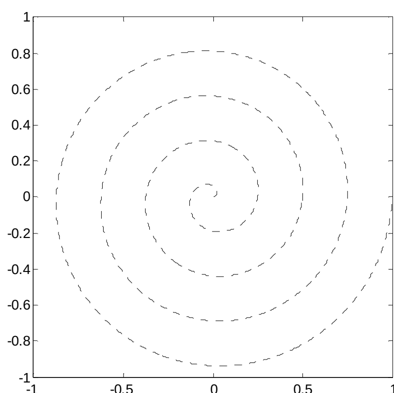


图 4-5 spirallength.m 运行结果图

M 文件函数参数如表 4-5 所示。

表 4-5 M 文件函数参数指令集

数据格式	说　　明
nargin	表示一个变量，指定调用函数所带参数的个数
nargout	表示一个变量，指定调用函数所返回的参数的个数
inputname(x)	返回输入表上数值 *x* 所在位置的输入参数变量的名字。如果用一个表达式代替已命名的参数，则返回一个空字符串
errorstr= nargchk(min, max,number)	功能为控制函数的输入参数的个数。参数 number 是由 nargin 指定的输入参数的个数。如果 number 的值超过 min 到 max 的区间范围，则系统将返回一个错误字符串 errorstr，否则将返回一个空矩阵
varargin	表示函数可带有任意多个输入参数的细胞矩阵
varargout	表示函数可带有任意多个输出参数的细胞矩阵

4.5.4　匿名函数、子函数、私有函数与私有目录

1．匿名函数

匿名函数没有函数名，也不是函数 M 文件，只包含一个表达式和输入/输出参数。用户可以在命令行窗口中输入代码，创建匿名函数。匿名函数的创建方法为：

```
f = @(input1,input2,…) expression
```

f 为创建的函数句柄。函数句柄是一种间接访问函数的途径，可以使用户调用函数过程变得简单，减少了程序设计中的繁杂，而且可以在执行函数调用过程中保存相关信息。

例 4-18，当给定实数 x 、y 的具体数值后要求计算表达式 x^y+3xy 的结果，可以通过创建匿名函数的方式来解决。

在命令行窗口中输入下述命令：

```
 Fxy = @(x,y) x.^y + 3*x*y
```

MATLAB 将创建一个名为 Fxy 的函数句柄：

```
Fxy = @(x,y)x.^y+3*x*y
```

调用 whos 函数，可以查看变量 Fxy 的信息：

```
 whos Fxy
  Name     Size           Bytes  Class           Attributes
  Fxy      1x1               16  function_handle
```

分别求当 x=2、y=5 及 x=1、y=9 时表达式的值：

```
 Fxy(2,5)
ans =   62
 Fxy(1,9)
ans =   28
```

2．子函数

在 MATLAB 中，多个函数的代码可以同时写到一个 M 函数文件中。其中，出现的第一个函数称为主函数（Primary Function），该文件中的其他函数称为子函数（Sub Function）。保存时所用的函数文件名应当与主函数定义名相同，外部程序只能对主函数进行调用。

子函数的书写规范有如下几条：

（1）每个子函数的第一行是其函数声明行。

（2）在 M 函数文件中，主函数的位置不能改变，但是多个子函数的排列顺序可以任意改变。

（3）子函数只能被处于同一 M 文件中的主函数或其他子函数调用。

（4）在 M 函数文件中，任何指令通过“名称”对函数进行调用时，子函数的优先级仅次于 MATLAB 内置函数。

（5）同一 M 文件的主函数、子函数的工作区都是彼此独立的。各个函数间的信息传递可以通过输入/输出变量、全局变量或跨空间指令来实现。

（6）help、lookfor 等帮助指令都不能显示一个 M 文件中的子函数的任何相关信息。

例 4-19，M 文件中的子函数示例。

创建 M 文件并命名为 ex4_19.m。

利用 M 文件编辑器，在 ex4_19.m 文件中写入：

```
function F = ex4_19(n)
A = 1; w = 2; phi = pi/2;
signal = createsig(A,w,phi);
F = signal.^n;
%---------subfunction---------
function signal = createsig(A,w,phi)
x = 0: pi/100 : pi*2;
signal = A * sin(w*x+phi);
```

3. 私有函数与私有目录

所谓私有函数，是指位于私有目录 private 下的 M 函数文件，它的主要性质有如下几条：

（1）私有函数的构造与普通 M 函数完全相同。

（2）关于私有函数的调用：私有函数只能被 private 直接父目录下的 M 文件所调用，而不能被其他目录下的任何 M 文件或 MATLAB 指令窗中的命令所调用。

（3）在 M 文件中，任何指令通过“名称”对函数进行调用时，私有函数的优先级仅次于 MATLAB 内置函数和子函数。

（4）help、lookfor 等帮助指令都不能显示一个私有函数文件的任何相关信息。

4.5.5 重载函数

重载是计算机编程中非常重要的概念，经常用于处理功能类似但变量属性不同的函数。例如实现两个相同的计算功能，输入的变量数量相同，不同的是其中一个输入变量类型为双精度浮点类型，另一个输入变量类型为整型。这时，用户就可以编写两个同名函数，分别处理这两种不同情况。当用户实际调用函数时，MATLAB 就会根据实际传递的变量类型选择执行哪一个函数。

MATLAB 的内置函数中就有许多重载函数，放置在不同的文件路径下，文件夹通常命名为“@+代表 MATLAB 数据类型的字符”。例如@int16 路径下的重载函数的输入变量应为 16 位整型变量，而@double 路径下的重载函数的输入变量应为双精度浮点类型。

4.5.6 eval 和 feval 函数

1. eval 函数

eval 函数可以与文本变量一起使用，实现有力的文本宏工具。其调用格式如下：

- eval(s)：该指令的功能为使用 MATLAB 的注释器求表达式的值或执行包含文本字符串 s 的语句。

例 4-20，eval 函数的简单运用示例。

本例共展示了利用 eval 函数分别计算 4 种不同类型的语句字符串，分别是：

- “表达式”字符串。

- “指令语句”字符串。
- “备选指令语句”字符串。
- “组合”字符串。

现示例如下：

（1）创建 M 文件并命名为 eval_exp1.m。

利用 M 文件编辑器，在 eval_exp1.m 文件中写入：

```
Array = 1:5;
String = '[Array*2; Array/2; 2.^Array]';
Output = eval(String)
```

输出结果：

```
Output =
   2.0000    4.0000    6.0000    8.0000   10.0000
   0.5000    1.0000    1.5000    2.0000    2.5000
   2.0000    4.0000    8.0000   16.0000   32.0000
```

（2）创建 M 文件并命名为 eval_exp2.m。

利用 M 文件编辑器，在 eval_exp2.m 文件中写入：

```
theta = pi;
eval('Output = exp(sin(theta))');
who
```

运行 M 文件，得到结果：

```
Output =
   1.0000
Your variables are:
Array   Output  String  Y      t      theta
```

（3）创建 M 文件并命名为 eval_exp3.m。

利用 M 文件编辑器，在 eval_exp3.m 文件中写入：

```
Matrix = magic(3)
Array = eval('Matrix(5,:)','Matrix(3,:)')
errmessage=lasterr
```

运行 M 文件，得到结果：

```
Matrix =
    8    1    6
    3    5    7
    4    9    2
Array =
    4    9    2
errmessage =
Index exceeds matrix dimensions.
```

（4）创建 M 文件并命名为 eval_exp4.m。

利用 M 文件编辑器，在 eval_exp4.m 文件中写入：

```
Expression = {'zeros','ones','rand','magic'};
Num = 2;
Output = [];
for i=1:length(Expression)
   Output = [Output eval([Expression{i},'(',num2str(Num),')'])];
```

```
end
Output
```

运行 M 文件，得到结果：

```
Output =   0       0    1.0000    1.0000    0.0318    0.0462    1.0000    3.0000
           0       0    1.0000    1.0000    0.2769    0.0971    4.0000    2.0000
```

2．feval 函数

feval 函数的具体句法形式如下：

```
[y1, y2, …] = feval('FN', arg1, arg2, …)
```

该指令的功能为用变量 arg1,arg2,…来执行 FN 函数指定的计算。

说明：

（1）在此 FN 为函数名。

（2）在 eval 函数与 feval 函数通用的情况下（使用这两个函数均可以解决问题），feval 函数的运行效率比 eval 函数高。

（3）feval 函数主要用来构造“泛函”型 M 函数文件。

例 4-21， feval 函数的简单运用示例。

（1）示例说明：feval 和 eval 函数运行区别之一是 feval 函数的 FN 不可以是表达式。

创建 M 文件并命名为 feval_exp1.m。

利用 M 文件编辑器，在 feval_exp1.m 文件中写入：

```
Array = 1:5;
String = '[Array*2; Array/2; 2.^Array]';
Outpute = eval(String)                          %使用 eval 函数运行表达式
Outputf = feval(String)                         %使用 feval 函数运行表达式
```

运行 M 文件，得到结果：

```
Outpute =
   2.0000    4.0000    6.0000    8.0000   10.0000
   0.5000    1.0000    1.5000    2.0000    2.5000
   2.0000    4.0000    8.0000   16.0000   32.0000
Invalid function name '[Array*2; Array/2; 2.^Array]'
```

（2）示例说明：feval 函数中的 FN 只接收函数名，不能接收表达式。

创建 M 文件并命名为 feval_exp2.m。

利用 M 文件编辑器，在 feval_exp2.m 文件中写入：

```
j = sqrt(-1);
Z = exp(j*(-pi:pi/100:pi));
eval('plot(Z)');
set(gcf,'units','normalized','position',[0.2,0.3,0.2,0.2])
title('Results by eval');axis('square')
figure
set(gcf,'units','normalized','position',[0.2,0.3,0.2,0.2])
feval('plot',Z);
title('Results by feval');axis('square')
```

运行 M 文件，结果如图 4-6 所示。

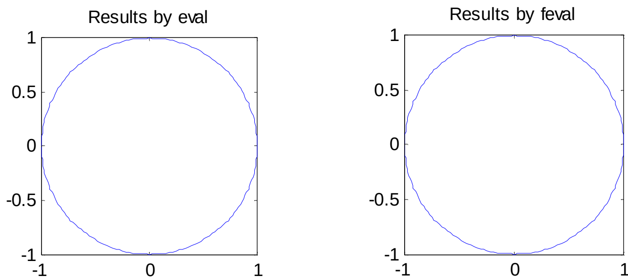


图 4-6　feval_exp2.m 文件的运行结果

4.5.7　内联函数

内联函数（Inline Function）的属性和编写方式与普通函数文件相同，但相对来说，内联函数的创建简单得多。其调用格式如下：

- inline('CE')：其功能为把字符串表达式“CE”转化为输入变量自动生成的内联函数。本语句将自动对字符串 CE 进行辨识，其中除了“预定义变量名”（例如圆周率 pi）、“常用函数名”（例如 sin、rand 等），其他由字母和数字组成的连续字符辨识为变量，连续字符后紧接左括号的，也不会被识别为变量，例如 array(1)。
- inline('CE', arg1, arg2, …)：其功能为把字符串表达式“CE”转换为 arg1、arg2 等指定的输入变量的内联函数。本语句创建的内联函数最为可靠，输入变量的字符串用户可以随意改变，但是由于输入变量已经规定，因此生成的内联函数不会出现辨识失误等错误。
- inline('CE', n)：其功能为把字符串表达式“CE”转化为 n 个指定的输入变量的内联函数。本语句对输入变量的字符是有限制的，其字符只能是 $x,P_1,\cdots,P_n$ 等，其中 P 一定为大写字母。

说明：

（1）字符串 CE 中不能包含赋值符号“=”。

（2）内联函数是沟通 eval 和 feval 两个函数的桥梁，只要是 eval 可以操作的表达式，都可以通过 inline 指令转化为内联函数，这样，内联函数总是可以被 feval 调用。MATLAB 中的许多内置函数就是通过被转换为内联函数，从而具备了根据被处理的方式不同而变换不同函数形式的能力。

MATLAB 中关于内联函数的属性的相关指令如表 4-6 所示，读者可以根据需要使用。

表 4-6　内联函数属性指令集

指令句法	功　　能
class(inline_fun)	提供内联函数的类型
char(inline_fun)	提供内联函数的计算公式
argnames(inline_fun)	提供内联函数的输入变量
vectorize(inline_fun)	使内联函数适用于数组运算的规则

例 4-22，内联函数的简单运用示例。

（1）示例说明：内联函数的第一种创建格式是使内联函数适用于“数组运算”。

在命令行窗口中输入：

```
Fun1=inline('mod(12,5)')
```

输出结果：

```
Fun1 =   内联函数:     Fun1(x) = mod(12,5)
```

在命令行窗口中输入：

```
Fun2=vectorize(Fun1)
```

输出结果：

```
Fun2 =   内联函数:     Fun2(x) = mod(12,5)
```

在命令行窗口中输入：

```
Fun3=char(Fun2)
```

输出结果：

```
Fun3 = mod(12,5)
```

（2）示例说明：第一种内联函数创建格式的缺陷在于不能使用多标量构成的向量进行赋值，而使用第二种内联函数创建格式则可以。

在命令行窗口中输入：

```
Fun4 = inline('m*exp(n(1))*cos(n(2))'), Fun4(1,[-1,pi/2])
```

输出结果：

```
Fun4 =

    内联函数:
    Fun4(m) = m*exp(n(1))*cos(n(2))

错误使用 inline/subsref (line 14)
内联函数的输入数目太多。
```

在命令行窗口中输入：

```
Fun5 = inline('m*exp(n(1))*cos(n(2))','m','n'), Fun5(1,[-1,pi/2])
```

输出结果：

```
Fun5 =
    内联函数:
    Fun5(m,n) = m*exp(n(1))*cos(n(2))
ans =
2.2526e-017
```

（3）示例说明：产生向量输入、向量输出的内联函数。

在命令行窗口中输入：

```
y = inline('[3*x(1)*x(2)^3;sin(x(2))]')
```

输出结果：

```
y =
    内联函数:
    y(x) = [3*x(1)*x(2)^3;sin(x(2))]
```

在命令行窗口中输入：

```
Y = inline('[3*x(1)*x(2)^3;sin(x(2))]')
```

输出结果：

```
Y =
    内联函数:
    Y(x) = [3*x(1)*x(2)^3;sin(x(2))]
```

在命令行窗口中输入：

```
argnames(Y)
```

输出结果：

```
ans =    'x'
```

在命令行窗口中输入：

```
x=[10,pi*5/6];y=Y(x)
```

输出结果：

```
y =
538.3034
0.5000
```

（4）示例说明：最简练的格式创建内联函数；内联函数可被 feval 指令调用。

在命令行窗口中输入：

```
Z=inline('floor(x)*sin(P1)*exp(P2^2)',2)
```

输出结果：

```
Z =   内联函数:    Z(x,P1,P2) = floor(x)*sin(P1)*exp(P2^2)
```

在命令行窗口中输入：

```
z = Z(2.3,pi/8,1.2), fz = feval(Z,2.3,pi/8,1.2)
```

输出结果：

```
z =   3.2304
fz =  3.2304
```

4.5.8　向量化和预分配

1. 向量化

要想让 MATLAB 最高速地工作，重要的是在 M 文件中把算法向量化。其他程序语言可以用 for 或 DO 循环，MATLAB 则可用向量或矩阵运算。下面的代码用于创立一个算法表：

```
x = 0.01;
for k = 1:1001
  y(k) = log10(x);
  x = x + 0.01;
end
```

同样代码的向量化翻译如下：

```
x =0 .01:0.01:10;
y = log10(x);
```

对于更复杂的代码，矩阵化选项不总是那么明显。当速度重要时，应该想办法把算法向量化。

2．预分配

若一条代码不能向量化，则可以通过预分配任何输出结果已保存其中的向量或数组以加快 for 循环。例如，下面的代码用 zeros 函数把 for 循环产生的向量预分配，这使得 for 循环的执行速度显著加快。

```
r = zeros(32,1);
for n = 1:32
   r(n) = rank(magic(n));
end
```

上例中若没有使用预分配，MATLAB 的注释器利用每次循环扩大 ***r*** 向量。向量预分配排除了该步骤以使执行加快。

4.5.9　函数的函数

一种以标量为变量的非线性函数称为“函数的函数”，即以函数名为自变量的函数。这类函数包括求零点、最优化、求积分和常微分方程等。

MATLAB 通过 M 文件的函数表示该非线性函数。

例如，下例为一个简化的 humps 函数（humps 函数可在路径 MATLAB\demos 下获得）。

例 4-23，函数的函数简单运用示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
a = 0:0.002:1;
b = humps(a);
plot(a,b)                                           %作出图像
function b = humps
b = 1./((x-.3).^2 + .01) + 1./((x-.9).^2 + .04) - 6;    %在区间[0,1]求此函数的值
```

运行文件，输出图像如图 4-7 所示。

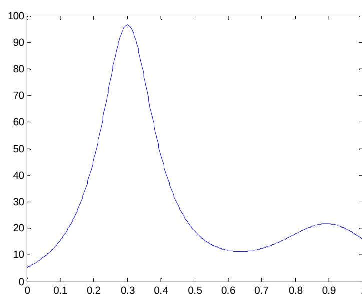


图 4-7　humps1.m 运行结果

图像表明函数在 *x*=0.6 附近有局部最小值。接下来用函数 fminsearch 可以求出局部最小值及此时 *x* 的值。函数 fminsearch 第一个参数是函数句柄，第二个参数是此时 *x* 的近似值。

在命令行窗口中输入：

```
p = fminsearch(@humps,.5)
```

输出结果：

```
p =
   0.6370
```

在命令行窗口中输入：

```
humps(p)                    %求出此局部最小值
```

输出结果：

```
ans =
   11.2528
```

4.5.10　P 码文件

一个 M 文件首次被调用(包括在 M 文件编辑器中被打开或者在命令行窗口运行文件名）时，MATLAB 将首先对该 M 文件进行语法分析，并把生成的相应内部伪代码（Psedocode，P 码）文件存放在内存中。

当 M 文件再次被调用时，将直接调用该 M 文件在内存中的 P 码文件，而不会再对原 M 文件进行重复的语法分析。需要注意的是，MATLAB 的分析器（Parser）总是把 M 文件连同在该 M 文件中被调用的所有函数文件一起转变成 P 码文件。

P 码文件与原码文件具有相同的文件名，但是其扩展名为“.p”；P 码文件的运行速度是高于原码文件的，但对于小规模的文件而言，用户一般体会不到这种速度上的差异。

在 MATLAB 环境中，假如存在同名的 P 码文件和原码文件，那么当该文件名被调用时，被执行的一定是 P 码文件。

P 码文件并不是仅当 M 文件被调用时才能生成的，用户也可以使用 MATLAB 中的内设指令在 M 文件中生成 P 码文件，其调用格式如下：

- pcode FunName：该指令的功能为在当前文件夹上生成 FunName.p。
- pcode FunName –inplace：该指令的功能为在 Filename.m 所在的目录上生成 FunName.p。

说明：P 码文件相对于原码文件来说有以下两个优点。

（1）运行速度快。

（2）P 码文件中的数据为二进制保存，阅读困难，增大了程序的保密性。

MATLAB 中还内置了 P 码文件的相关操作指令，如表 4-7 所示。

表 4-7　P 码文件的相关操作指令

指 令 名	功　能
inmen	罗列内存中的所有 P 码文件的文件名
clear FunName	清除内存中名为 FunName.p 的 P 码文件
clear functions	清除内存中所有的 P 码文件

例 4-24，查询内存中所有的 P 码文件名，并清除指定名称的 P 码文件。

首先调用 inmem 函数，查询当前内存中所有的 P 码文件名，得到如下结果：

```
inmem
ans =
    'MATLABrc'
    'pathdef'
    'userpath'
    'ispc'
    'filesep'
    'pwd'
    'usejava'
    'initializeMATLABRoot'
    'opaque.char'
'hgrc'
…
    'helpProcess.getDemoTopic'
    'helpProcess.displayHelp'
```

然后使用 clear 函数，清除名为 pwd 的 P 码文件：

```
clear pwd
```

再次调用 inmem 函数，查看当前内存中所有的 P 码文件，发现名为 pwd 的 P 码文件不存在，得到如下结果：

```
inmem
ans =
    'MATLABrc'
    'pathdef'
    'userpath'
    'ispc'
    'filesep'
    'usejava'
    'initializeMATLABRoot'
    'opaque.char'
'hgrc'
…
    'helpProcess.getDemoTopic'
    'helpProcess.displayHelp'
```

4.6　M 文件中变量的检测与传递

M 文件中不同文件之间数据的传递是以变量为载体来实现的，数据的保存和中转则都是以空间为载体来实现的。因此，M 文件中变量的检测与传递是检验运算关系和运算正确与否的有力保障。

4.6.1　输入/输出变量检测指令

MATLAB 中内置了相关的输入/输出变量的检测指令，如表 4-8 所示。

表 4-8　P 码文件的相关操作指令

指 令 名	功　能
Nargin	在函数体内获得实际的输入变量
Nargout	在函数体内获得实际的输出变量
nargin('fun')	获取 fun 指定的函数的标称输入变量数量
nargout('fun')	获取 fun 指定的函数的标称输出变量数量
inputname(n)	在函数体内给出第 *n* 个输入变量的实际调用变量名

说明：

（1）在函数体内使用 nargin 和 nargout 的目的是与程序流控制指令配合，对于不同数目的输入/输出变量，函数可以完成不同的任务。

（2）nargin、nargout 和 inputname 本身都是函数，而不是变量，因此用户不能使用赋值指令对其进行处理。

4.6.2　“可变数量”输入/输出变量

MATLAB 中的许多指令或函数具有输入变量可以是任意多个的特点。

以 MATLAB 中的绘图函数 plot（在第 6 章中将会大量使用该函数）为例来说明 MATLAB 中的“可变数量”输入/输出变量现象。例如调用以下句式：

```
plot(a, b, 'PropertyName1', 'PropertyName1', 'PropertyValue1',
  'PropertyName2',…)
```

在上述句式中，plot 函数允许使用任意多的“属性名/属性值”来精确规定绘图的规范。

在 MATLAB 中还有一些函数，它们也都具有“可变数量”输入/输出变量数量的性质。为了使得用户自主编写的程序也可以具备这条性质，MATLAB 中内置了以下函数来实现这个功能，如表 4-9 所示。

表 4-9　可变数量输入/输出变量指令

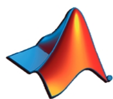
指 令 名	功　能
varargin	“可变数量”输入变量列表
varargout	“可变数量”输出变量列表

说明：

（1）在编写 M 函数文件时，函数声明行中的“可变数量”输入/输出变量必须被放置在“正常”变量之后。

（2）在编写 M 函数文件时，varargin 的每个元胞应当作为一个普通的输入变量来对待。

（3）varargin 与 varargout 的工作机理类似，差别仅在于 varargout 承载的是输出变量和输入变量之间的配置关系。以 varargin 为例，该函数的工作机理：①varargin 本身的数据类型为一个元胞数组。②在 M 文件函数被调用时，输入变量按先后顺序依次对应分配给 M 函数文件的输入变量的列表中那些被明确定义的普通输入变量，然后把剩余的输入变量依次分配到 varargin 元胞数组的元胞中。因此，varargin 元胞数组的长度取决于分配到的输入变量数量。③所谓“可变数量”，指的是 varargin 的长度随着分配到的输入变量数量而改变。



例 4-25，可变长度数据量使用示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
function varargout = spirallength(d, n, varargin)
%SPIRAL 画出螺旋线或螺旋条带
Nin = length(varargin) + 1;
% error(nargchk(1, Nin, nargin))
if nargout > 1
  error('Too many output arguments!!!')
end
j = sqrt(-1);
phi = 0: pi/20 : n*2*pi;
amp = 0: d/40 : n*d;
spir = amp .* exp(j*phi);
if nargout==0
   switch Nin
   case 1
      plot(spir,'b')
   case 2
      d1=varargin{1};
      amp1 = (0: d/40 : n*d) + d1; spir1 = amp1 .* exp(j*phi);
      plot(spir,'b');hold on;plot(spir1,'b');hold off
      otherwise
      d1=varargin{1};
      amp1 = (0: d/40 : n*d) + d1; spir1 = amp1 .* exp(j*phi);
      plot(spir,varargin{2:end});hold on;plot(spir1,varargin{2:end});
   end;
   axis('square')
else
   phi0 = 0: pi/1000 : n*2*pi;
   amp0 = 0: d/2000 : n*d;
   spir0 = amp0 .* exp(j*phi0);
   varargout{1} = sum(abs(diff(spir0)));
   if Nin>1
      d1=varargin{1};
      amp1 = (0: d/2000 : n*d) + d1; spir1 = amp1 .* exp(j*phi);
      varargout{2} = sum(abs(diff(spir1)));
   end;
end
```

在命令行窗口中输入：

```
subplot(1,3,1), spirallength(2,2,1)
subplot(1,3,2), spirallength(2,2,1,'Marker','o')
subplot(1,3,3), spirallength(2,2,1,'r--', 'LineWidth',2)
```

输出结果如图 4-8 所示。

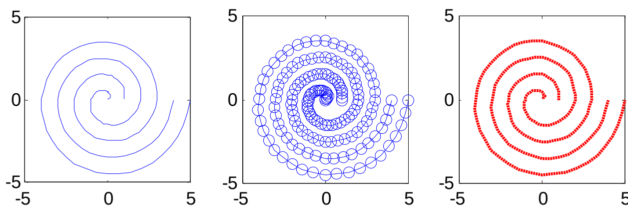


图 4-8　spirallength.m 运行结果

4.6.3　跨空间变量传递和赋值

1．跨空间变量传递

在 MATLAB 中，不同工作区之间的数据传递是通过变量来实现的，主要形式有两种：函数的输入/输出变量和全局变量。本节将对除此之外的第 3 个数据传递渠道——跨空间计算表达式的值进行介绍。其具体的调用格式如下：

- evalin('工作区', 'expression')：该指令的功能为跨空间计算字符串表达式的值。

说明：

（1）“工作区”的可取值有两个，分别是“base”和“caller”。

（2）工作机理为：①当“工作区”为“base”，表达式计算 eval("expression")时，将从基本工作区获得变量值；②当“工作区”为“caller”，表达式计算 eval("expression")时，将从主调函数基本工作区获得变量值。主调函数是相对于被调函数而言的，被调函数是指 evalin 所在的函数。

- evalin('工作区', 'expression1', 'expression2')：该指令的功能为跨空间计算替代字符串表达式的值。

说明：

（1）“工作区”的可取值有两个，分别是“base”和“caller”。

（2）工作机理为先从所在函数空间获取变量值，用 eval("expression1")计算原字符串表达式，如果该计算失败，则从“工作区”指定的基本工作区或主调函数工作区获得变量值，再通过 eval("expression2")计算替代字符串表达式的值。

例 4-26，跨空间变量传递示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
clear all; close all;
n = 6;                          %n=6，作出的图片为六瓣的花
j = sqrt(-1);
phi = 0 : pi/(20*n) : 2*pi;
amp = 0;
for i = 1 : n
   amp = [amp 1/20:1/20:1 19/20:-1/20:0];
end
string = {'base', 'caller', 'self'};       %在本例中，字符串为“base”时，
    调用基本工作区
                                %字符串为“caller”时，调用函数空间
                                %字符串为“self”时，调用子函数空间
for i=1:3
   y = Flower(5, string{i});      %n=5，作出的图片为 5 瓣的花
   subplot(1, 3, i)
   plot(y,'r','LineWidth',4)
   axis('square')
end
```

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
function y1=Flower(n,s)
j = sqrt(-1);
```

```
phi = 0 : pi/(20*n) : 2*pi;
amp = 0;
for i = 1 : n
   amp = [amp 1/20:1/20:1 19/20:-1/20:0];
end
y1=subflower(4,s);                    %n=4，作出的图片为 4 瓣的花
%------------ subfunction -------------
function y2=subflower(n,s)
j = sqrt(-1);
phi = 0 : pi/(20*n) : 2*pi;
amp = 0;
for i = 1 : n
   amp = [amp 1/20:1/20:1 19/20:-1/20:0];
end
func='amp.*exp(j*phi)';
switch s
case {'base','caller'}
  y2=evalin(s,func);
case 'self'
  y2=eval(func);
end
```

运行文件，输出图像如图 4-9 所示。

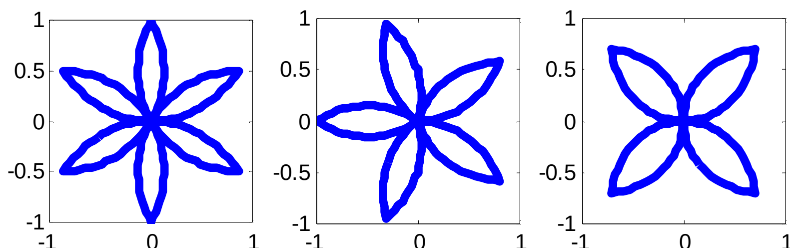


图 4-9　flowers.m 运行结果

2．跨空间变量赋值

实现不同工作区之间变量传递的第 4 个方式是跨空间变量赋值，其具体的调用格式如下：

- assignin('工作区', 'VN', x)：该指令的功能是为跨空间变量 VN 赋值，把当前工作区中的变量 x 的值赋值给指定的“工作区”中的指定变量 VN。assignin 的使用机理与跨空间变量传递的机理相同，因此此处不再举例说明，只举例说明使用 assignin 指令的注意事项。

例 4-27，assignin 运作机理示例。

在命令行窗口中输入：

```
assignin('base', 'Num', 0)
Num
```

输出结果：

```
Num =    0
```

在命令行窗口中输入：

```
Array = 1:10;
assignin('base', 'Array(4:7)', Num)
```

输出结果：

```
错误使用 assignin
ASSIGNIN 中的变量名称 "Array(4:7)" 无效。
```

在命令行窗口中输入：

```
evalin('base', 'Array(4:7) = Num')
```

输出结果：

```
X =123    0    0    0    0    8910
```

4.7　MATLAB 程序的调试

对于程序设计人员而言，程序试运行的过程中出现各种各样的问题（BUG）都是不可避免的，特别是对于那些规模较大、多组人员共同开发完成的大型应用程序更是如此。因此，熟练掌握程序的调试方法是一个合格的程序设计者必备的基本素质。

本节首先介绍程序出错的两类基本根源——语法错误和逻辑错误；在此基础上讲述程序调试的基本概念；最后介绍 MATLAB 中程序调试的两种基本方法——直接调试法和工具调试法。

4.7.1　程序调试的基本概念

一般而言，程序出现的错误可以分为以下三大类。

1. 语法错误

语法错误是指变量名的命名不符合 MATLAB 的规则、函数名的误写、函数的调用格式发生错误、标点符号的缺漏、循环中遗漏了“end”等情况。

2. 逻辑错误

逻辑错误主要表现在程序运行后，得到的结果与预期设想的不一致，这就有可能是出现了逻辑错误。通常出现逻辑错误的程序都能正常运行，系统不会给出提示信息，所以很难发现。例如在循环过程中没有设置跳出循环的条件，从而导致程序陷入死循环。要发现和改正逻辑错误，需要仔细阅读和分析程序。

3. 异常

异常是指程序执行过程中由于不满足前置条件或后置条件而造成的程序执行错误。例如等待读取的数据文件不在当前的搜索路径内等。

综上所述，程序调试就是将编制的程序投入实际运行前，用手工或编译程序等方法进行测试、修正语法错误和逻辑错误、解决异常状况的过程，这是保证计算机信息系统正确性的必不可少的步骤。有时调试工作所占用的时间甚至超过程序设计、代码编写所用的时间总和。

调试的目的在于发现其中的错误并及时纠正，所以在调试时应想方设法使程序的各个部分都投入运行，力图找出所有错误。错误多少与程序质量有关。即使调试通过也不能证明系统绝对无误，只不过说明各模块、各子系统的功能和运行情况正常，相互之间连接无误。程序在交付最终用户使用以后，在系统的维护阶段仍有可能发现少量错误，这也是正常的。

MATLAB 是一种边解释边执行的程序语言，特别是良好的所见即所得的特性，为程序的调试带来了诸多的方便。MATLAB 不仅提供了一系列的调试函数用于调试，而且在 M 文件编辑器中集成了程序调试器。通过使用调试器，用户可以完成调试工作。

4.7.2 直接调试法

对于简单的程序（例如只有单一模块组成的程序），直接调试法是一种简便快捷的方法。直接调试法的基本方法大致有如下几种：

（1）通过分析，将重点怀疑语句后的分号删除，将结果显示出来，然后与预期值比较，从而快速地判断程序执行到该处时是否发生了错误。

（2）在适当的位置添加输出变量值的语句。

（3）在程序的适当位置添加 keyboard 命令。当程序执行到该处时将暂停，并显示提示符，用户可以查看或变更工作区中显示的各个变量的值。在提示符后输入 return 指令可以继续执行原文件。

（4）调试程序时，可以利用注释符号“%”屏蔽函数声明行，并定义输入变量的值，以脚本 M 文件的方式执行程序，可以方便地查看中间变量，从而有利于找出相应的错误之处。

4.7.3 使用调试函数进行调试

MATLAB 提供了一系列的函数可供用户在调试程序时使用。这些函数主要用于程序执行过程相关的显示、执行中断、设置断点、执行单步操作等。在 MATLAB 命令行窗口中输入下述命令：

```
help debug
```

系统输出调试函数及其用途简介。这些函数都是以“db”开头的：

```
debug List MATLAB debugging functions
   dbstop     - Set breakpoint.                  % 设置断点
   dbclear    - Remove breakpoint.               % 清除断点
   dbcont     - Resume execution.                % 重新执行
   dbdown     - Change local 工作区 context.      % 变更本地工作区上下文
   dbmex      - Enable MEX-file debugging.       % 使 MEX 文件调试有效
   dbstack    - List who called whom.            % 列出函数调用关系
   dbstatus   - List all breakpoints.            % 列出所有断点
   dbstep     - Execute one or more lines.       % 单步或多步执行
   dbtype     - List file with line numbers.     % 列出 M 文件（包括行号）
   dbup       - Change local 工作区 context.      % 变更本地工作区上下文
```

```
dbquit    - Quit debug mode.                           % 退出调试模式
```

下面将分别介绍这些调试函数。

1．断点设置函数 dbstop

在程序的适当位置设置断点，使得 MATLAB 在断点前停止执行，用户可以检查各个局部变量的值。在命令行窗口中输入下述命令：

```
help dbstop
```

MATLAB 即列出 dbstop 函数的相关调用方法：

```
（1）dbstop in FILESPEC at LINENO
（2）dbstop in FILESPEC at LINENO@
（3）dbstop in FILESPEC at LINENO@N
（4）dbstop in FILESPEC at SUBFUN
（5）dbstop in FILESPEC
（6）dbstop in FILESPEC at LINENO if EXPRESSION
（7）dbstop in FILESPEC at LINENO@ if EXPRESSION
（8）dbstop in FILESPEC at LINENO@N if EXPRESSION
（9）dbstop in FILESPEC at SUBFUN if EXPRESSION
（10）dbstop in FILESPEC if EXPRESSION
（11）dbstop if error
（12）dbstop if caught error
（13）dbstop if warning
（14）dbstop if naninf  or  dbstop if infnan
（15）dbstop if error IDENTIFIER
（16）dbstop if caught error IDENTIFIER
（17）dbstop if warning IDENTIFIER
```

dbstop 函数常用的调用格式主要有以下几种：

- dbstop in FILESPEC at LINENO：在文件名为 FILESPEC 的 M 文件(该 M 文件必须在 MATLAB 搜索路径或当前文件夹中）的第 LINENO 行设置断点。执行该命令后，当程序运行到 FILESPEC 的第 LINENO 行时，可暂时中止该 M 文件的执行，并进入调试模式。如果行号为 LINENO 的语句为非执行语句，则在停止执行的同时，在该行号的下一行可执行语句前设置断点。此时，用户可以使用各种调试工具，查看工作区的变量的内容。
- dbstop in FILESPEC at SUBFUN：执行该命令后，当程序执行到子程序 SUBFUN 时，暂时中止文件的执行并使 MATLAB 处于调试模式。
- dbstop in FILESPEC：在文件名为 FILESPEC 的 M 文件的第一个可执行语句前设置断点。执行该命令后，当程序运行到 FILESPEC 的第一个可执行语句时，暂时中止 M 文件的执行，进入调试模式。
- dbstop if error：执行该命令后，当程序运行遇到错误时，终止 M 文件的执行，并停止在产生错误的那一行。这里的错误不包括 try…catch 语句中检测到的错误，用户在错误后重新开始程序的运行。
- dbstop if caught error：执行该命令后，在运行 M 文件的过程中，当 try…catch 语句检测到错误时，停止 M 文件的执行，用户可以恢复程序的运行。
- dbstop if warning：执行该命令后，在运行 M 文件的过程中，如果遇到警告，则终

止 M 文件的运行，并停在产生警告的那一行，MATLAB 进入调试状态。程序可以恢复运行。

- dbstop if naninf 或 dbstop if infnan：执行该命令后，当遇到无穷值或非数值时，终止 M 文件的执行。

2．断点清除函数 dbclear

dbclear 函数可以清除 dbstop 函数设置的断点。

在命令行窗口中输入下述命令：

```
help dbclear
```

MATLAB 即列出 dbclear 函数的相关调用方法：

```
（1）dbclear in FILESPEC at LINENO
（2）dbclear in FILESPEC at LINENO@
（3）dbclear in FILESPEC at LINENO@N
（4）dbclear in FILESPEC at SUBFUN
（5）dbclear in FILESPEC
（6）dbclear if ERROR
（7）dbclear if CAUGHT ERROR
（8）dbclear if WARNING
（9）dbclear if NANINF  or  dbclear if INFNAN
（10）dbclear if ERROR IDENTIFIER
（11）dbclear if CAUGHT ERROR IDENTIFIER
（12）dbclear if WARNING IDENTIFIER
（13）dbclear ALL
```

dbclear ALL 用来清除所有 M 文件中设置的所有断点；其余函数分别清除由 dbstop 函数相应的调用格式产生的各种断点，对照 dbstop 函数即可理解。

3．列出所有的断点函数 dbstatus

dbstatus 函数可以列出所有的断点，包括错误、警告、NAN 和 INF 等。该函数的调用格式如下：

- dbstatus FILESPEC：执行该命令后，MATLAB 列出文件名为 FILESPEC 的 M 文件中设置的所有断点。FILESPEC 必须是 M 文件的函数名称或有效的路径名称。
- S = dbstatus：该命令将返回一个 $m\times 1$ 的结构体数组 S，S 中存储了断点的相关信息。有以下字段。①name：函数名；②file：包含所有的 M 文件的文件名全称；③line：断点行号向量；④anonymous：存储了匿名函数内部所有断点位置的向量；⑤expression：与 line 中相对应的断点条件表达字符串，是一个单元向量；⑥cond：条件字符串，包括 error、caught error、warning 和 naninf；⑦identifier：是一个单元向量，存放根据条件字符串得到的 MATLAB 信息指示字符串。

4．恢复执行函数 dbcont

dbcont 函数从断点处恢复程序的执行，直到遇到程序的另一个断点或错误。

5．调用堆栈函数 dbstack

dbstack 函数用来显示 M 文件名和断点产生的行号、调用此 M 文件的名称和行号等，直到最高级的 M 文件函数。该函数的使用方法主要有以下几种。

- dbstack('-completenames')：该命令输出堆栈中每个函数的全名和在堆栈中函数包含的关系。
- [ST,I] = dbstack：该命令返回一个 m×1 的结构体数组 ST，ST 中存储了堆栈的相关信息。有以下字段：①file：函数出现的文件名，如果没有则为空；②name：文件中的函数名；③line：函数行号，同时该函数返回 I 用来记录当前工作区的索引。

6．执行一行或多行语句函数 dbstep

dbstep 函数可以执行 M 文件中的一行或多行语句。当执行完毕后，返回调试模式。如果在执行过程中遇到断点，程序将终止。该函数的使用方法有以下几种：

- Dbstep：该命令执行当前 M 文件中的下一个可执行语句。
- dbstep nlines：该命令执行当前 M 文件中由 nlines 指定数量的行数的可执行语句。
- dbstep IN：当执行下一个可执行语句时，如果其中包含对另外一个函数的调用，则此命令将从被调用的函数文件的第一个可执行语句开始执行。
- dbstep OUT：该命令执行函数剩余的部分，在离开函数时停止。

7. 列出 M 文件内容函数 dbtype

dbtype 函数可以列出 M 文件的内容，并在每行语句前加上行号，从而方便用户设置断点。该函数的使用方法有以下几种：

- dbtype FILESPEC：该命令列出文件名为 FILESPEC 的 M 文件的内容，并在每行语句前加上行号。
- dbtype FILESPEC RANGE：该命令列出文件名为 FILESPEC 的 M 文件及由参数 RANGE 指定的行号范围内的内容，并在每行语句前加上行号。例如 dbtype myfun.m 10:24。

8. 切换工作区函数 dbup 和 dbdown

dbup 和 dbdown 函数的使用方法：

- Dbup：该命令将当前工作区（断点处）切换到调用 M 文件的工作区。
- Dbdown：当遇到断点时，该命令将当前工作区切换到被调用的 M 文件的空间。

9. 退出调试模式函数 dbquit

dbquit 函数可以结束调试器并返回基本工作区，所有断点仍有效。该函数的使用方法主要有以下两种：

- dbquit：结束当前文件的调试。
- dbquit('all')：结束所有文件的调试。

4.7.4　工具调试法

本节介绍利用 MATLAB 的 M 文件编辑器（Editor）中集成的程序调试工具对程序进行调试。

单击工作界面中的“新建”图标，编辑器自动新建一个名为 Untitled（x）的 M 文件，此时，工具栏上的大部分按钮高亮显示，即为可用状态，如图 4-10 所示。

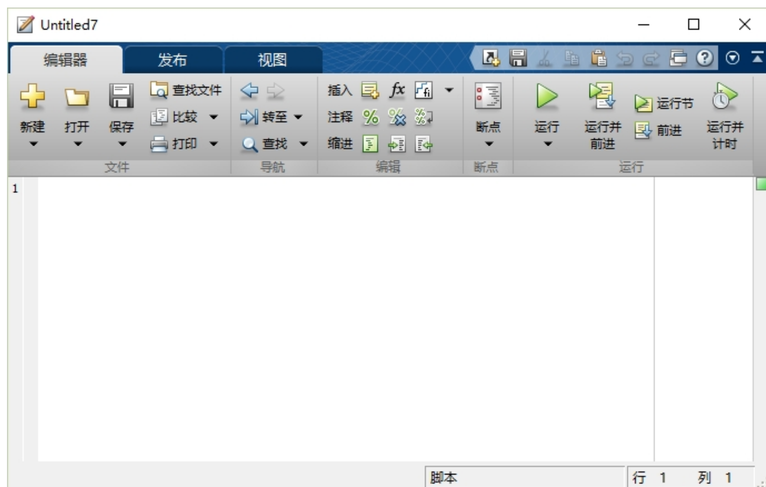


图 4-10　新建 M 文件

编辑器集成了各种程序调试命令，这些命令以按钮显示。这些按钮的功能与部分调试函数的功能是一样的。下面将分别介绍常见命令。

- 运行：保存并运行当前 M 文件，快捷键为 F5。
- 断点：设置或清除断点，快捷键为 F12，与调试函数中的 dbstop 和 dbclear 相对应。
- 全部清除断点：清除所有的断点，与调试函数中的 dbclear ALL 相对应。
- 运行并前进：连续执行，快捷键为 F5，与调试函数中的 dbcont 相对应。

4.7.5　程序的性能优化技术

实践证明，如果在开发程序时不重视性能的优化，最终虽然实现了功能上的要求，但程序的运行效率低下，也不能给用户带来很好的效益。因此，程序的性能优化是计算机程序开发过程中需要一直关注的重要因素。

本节将分别从程序的执行效率和内存的使用效率两个角度，介绍 MATLAB 程序的性能优化技术。

1. 程序的执行效率

要想提高程序运行效率，改进算法是最关键的。算法是影响程序运行效率的主要因素，在编写不同程序时要选择适当的算法。

例如求 $1+2+\mathbf{L}+99$，可以按照顺序依次加起来，也可以采用 $(1+99)+(2+98)+\mathbf{L}$ 的办法来计算，显然第二种方法更适合人们计算，它的效率比顺序相加快得多，甚至口算就能迅速得到答案。但对于计算机来说，第二种方法并不比第一种快，如果程序编制不当，反而会降低计算速度。因此，选择适当的算法是提高运行效率的关键。此外，影响程序运

行速度的地方都是在执行次数最多的地方，例如乘法和除法都是相当浪费 CPU 运算时间的运算。

综上所述，用户在使用 MATLAB 进行程序设计时，可以考虑以下一些常用方法来提高程序的执行效率：

- 尽可能地使用 load 函数及 save 函数，而不是文件 I/O 操作函数。
- 避免更改变量的数据类型或维数。如果确实需要这么做，可以预先创建一个新的变量。
- 尽可能地使用实数运算。因此，对于复数的运算可以转换为多个实数运算，由此提升效率。
- 尽可能地避免对实数和复数的相互赋值。
- 在进行逻辑运算时，采用&&等短路逻辑运算具有更高的效率。
- 代码向量化。这是因为 MATLAB 执行循环的效率比较低，因此，将诸如 for 循环和 while 循环转化为矩阵的按位运算，可以提高计算效率。
- 对不可避免且耗时很长的循环操作，可以尝试在 MEX 文件内实现。
- 在进行 for、while 等循环前，对于循环过程中不断变化的变量应预先分配足够大的数组，从而避免 MATLAB 频繁地进行变量数组重生成操作，提高运算速度。
- 尽可能地采用函数 M 文件而不是脚本 M 文件，因为函数 M 文件的执行效率要高于脚本 M 文件。

2. 内存优化技术

MATLAB 提供了一系列的函数帮助用户了解 MATLAB 对内存的使用情况，并进行相关的操作。比较常用的函数：

- whos 函数：用于查看当前的内存使用情况。
- clear 函数：该函数的调用格式如下：

```
clear variablename    %从内存中删除名称为 variablename 的变量
clear all             % 从内存中删除所有的变量
```

- save 函数：将指定的变量存入磁盘。
- load 函数：将 save 命令存入的变量载入内存。
- quit 函数：退出 MATLAB，并释放所有分配的内存。
- pack 函数：把内存中的变量存入磁盘，再用内存中的连续空间载回这些变量(不要在循环中使用）。

为了节约且更加有效地使用内存，用户在程序设计时应该注意以下几点：

（1）尽可能在函数开始处创建变量。

（2）避免生成大的中间变量，并删除不再需要的临时变量。

（3）当使用大的矩阵变量时，预先指定维数并分配好内存，避免每次临时扩充维数。

（4）当程序需要生成大量变量数据时，可以考虑将变量写到磁盘，然后清除这些变量。当需要这些变量时，重新从磁盘加载。

（5）当矩阵中数据极少时，将全矩阵转换为稀疏矩阵。

4.8 本章小结

本章主要分析了 MATLAB 编程中的基本概念，包括控制流、脚本、函数和 P 码文件等，这些都是使用 MATLAB 的基础，读者应当熟练掌握。掌握了这些知识，基本的编程已经没有问题了。关于 MATLAB 编程中的一些技巧，例如向量化和预分配等，初学者可以先了解概念，在以后的编程中慢慢体会。

第 5 章

数据可视化

知识要点

在科学的研究体系中，将数学公式与数据表现在图表中，是展示符号的具体物理含义及大量数据的内在联系和规律的科学有效的方法。

MATLAB 中可以绘制二维、三维和四维的数据图形，并且通过对图形的线型、颜色、标记、观察角度、坐标轴范围等属性的设置，将大量数据的内在联系及规律表现得更加细腻、完善。MATLAB 提供众多的设备用图表来显示向量和矩阵，同时包括注释和打印这些图表。

MATLAB 拥有大量简单、灵活、易用的二维和三维图形命令，并且用户可以在 MATLAB 程序中加入声音效果。本章将详细讲述 MATLAB 中的数据可视化技术。

学习要求	学习目标 知识点	了解	理解	应用	实践
	图形绘制的基本概念		√		
	二维图形绘制			√	√
	三维图形绘制		√		√
	四维图形绘制	√			



5.1 图形绘制

基于由浅入深的原则，本节将从最简单的平面上的点的表示入手，逐步深入，由离散数据的表示到连续数据的表示，使得读者掌握其中规律。

5.1.1 离散数据及离散函数

一个二元实数标量对(x_0, y_0)可以用平面上的点来表示，一个二元实数标量数组[(x_1, y_1) (x_2, y_2) … (x_n, y_n)]可以用平面上的一组点来表示。对于离散函数 $Y=f(X)$，当 X 为一维标量数组[x_1, x_2, … ,x_n]时，根据函数关系可以求出 Y 相应的一维标量[y_1, y_2, … ,y_n]。

当把这两个向量数组在直角坐标系中用点序列来表示时，就实现了离散函数的可视化。当然，这些图形上的离散序列所反映的只是 X 所限定的有限点上或是有限区间内的函数关系。应当注意的是，MATLAB 是无法实现对无限区间上的数据的可视化的。

例 5-1，离散数据和离散函数的可视化。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
X1 = [1 2 4 6 7 8 10 11 12 14 16 17 18 20];
Y1 = [1 2 4 6 7 8 10 10 8 7 6 4 2 1];       %生成两个一维实数数组
figure(1)
plot(X1, Y1, 'o', 'MarkerSize', 15)
X2 = 1:20;
Y2 = log(X2);                                %根据 log 函数生成两个一维实数数组
figure(2)
plot(X2, Y2, 'o', 'MarkerSize', 15)
```

运行 M 文件，结果如图 5-1 所示。

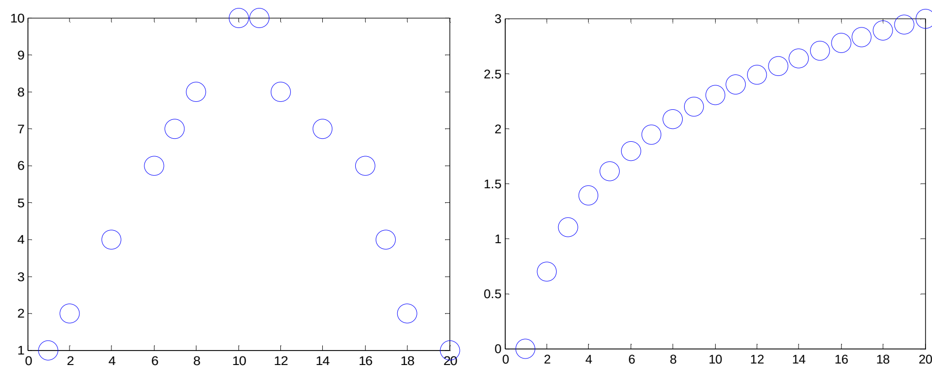


图 5-1 logfigure.m 运行结果

5.1.2 连续函数

在 MATLAB 中是无法画出真正的连续函数的，因此在实现连续函数的可视化时，首先必须将连续函数用在一组离散自变量上计算函数结果，然后将自变量数组和结果数组在图形

中表示出来。

当然，这些离散的点还是不能表现函数的连续性的。为了更形象地表现函数的规律及其连续变化，通常采用以下两种方法：

（1）对离散区间进行更细的划分，逐步趋近函数的连续变化特性，直到达到视觉上的连续效果。

（2）把每两个离散点用直线连接，以每两个离散点之间的直线来近似表示两点间的函数特性。

例 5-2，连续函数的可视化。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
X1 = (0:12)*pi/6; Y1 = cos(3*X1);
X2 = (0:360)*pi/180; Y2 = cos(3*X2);
figure(1)
subplot(2,2,1); plot(X1, Y1, 'o', 'MarkerSize', 3); xlim([0 2*pi])
subplot(2,2,2); plot(X1, Y1, 'LineWidth', 2); xlim([0 2*pi])
subplot(2,2,3); plot(X2, Y2, 'o', 'MarkerSize', 3); xlim([0 2*pi])
subplot(2,2,4); plot(X2, Y2, 'LineWidth', 2); xlim([0 2*pi])
```

运行 M 文件，结果如图 5-2 所示。

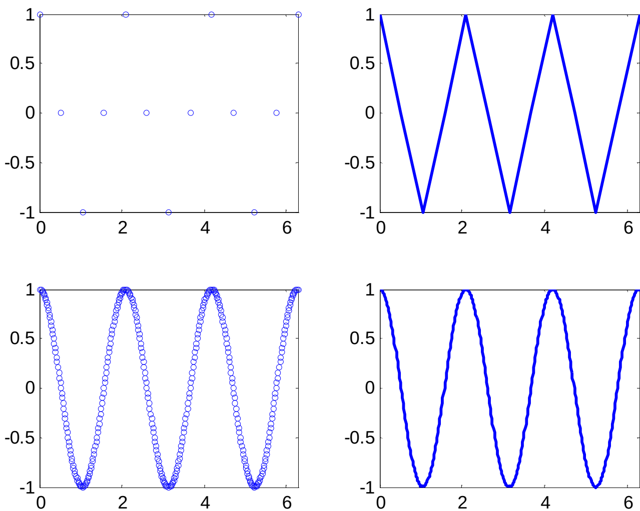


图 5-2　cosfigure.m 运行结果

5.1.3　图形绘制示例

例 5-3，设函数 $y = x + \sin x + e^{x}$，试利用 MATLAB 绘制该函数在 $x \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ 上的图像。

（1）准备图形数据。用户需要选定数据的范围，选择对应范围的自变量，计算相应的函数值。根据要求，在本实例中，需要在命令行窗口中输入下列命令：

```
x = -pi/2:0.01:pi/2;
 y = x + sin(x) + exp(x);
```

（2）使用 plot 函数绘制图形，即在命令行窗口中输入下列命令：

```
plot(x,y)
```

得到的结果如图 5-3 所示。

（3）为了更好地观察各个数据点的位置，给背景设置网格线，同时采用空心圆圈来标记数据点，并将曲线的颜色设置成红色。因此，在命令行窗口中输入：

```
plot(x,y,'-ro')
 grid on
```

得到的结果如图 5-4 所示。

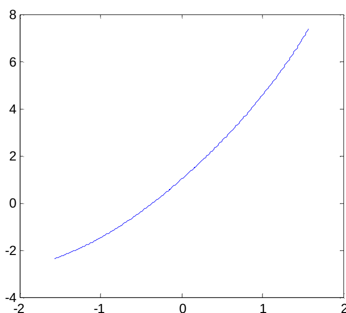


图 5-3　绘制的函数图像

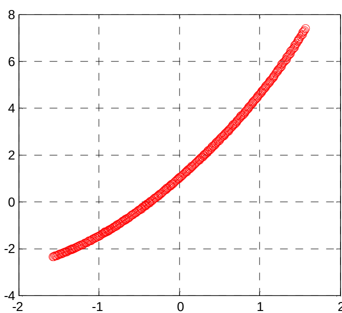


图 5-4　添加网格线，修改曲线样式

（4）给图形添加一些注释。为了进一步使图形具有可读性，用户还需要给图形添加一些注释，例如图形的名称、坐标轴的名称、图例及文字说明等。例如本实例给图形取名字为“y 的函数图像”；x 坐标轴和 y 坐标轴分别取名为“x”和“y”；图例设置为“$y=x+\sin x+e^x$”。因此，需要在命令行窗口中输入：

```
title('y 的函数图像');
xlabel('x');
ylabel('y');
legend('y=x+sinx+e^{x}');
```

得到的结果如图 5-5 所示。

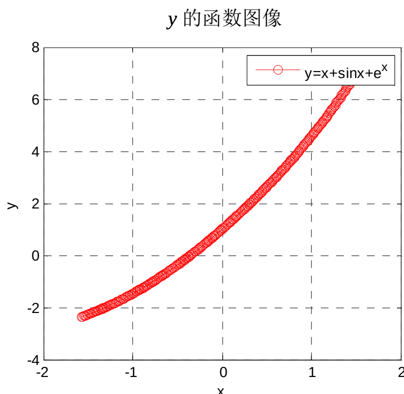


图 5-5　添加图形注释

（5）图形的输出。完成图形的绘制和编辑之后，用户可以将图形打印或者在图形窗口

的菜单栏中选择 File→Save As 命令，将图形保存成用户需要的格式。

5.1.4　图形绘制的基本步骤

通过 5.1.3 节的示例可以总结出，利用 MATLAB 绘制图形大致分为如下 7 个步骤：

（1）数据准备。主要工作是产生自变量采样向量，计算相应的函数值向量。

（2）选定图形窗口及子图位置。在默认情况下，MATLAB 系统绘制的图形为 figure.1、figure.2……

（3）调用绘图函数绘制图形，例如 plot 函数。

（4）设置坐标轴的范围、刻度及坐标网格。

（5）利用对象属性值或者图形窗口工具栏设置线型、标记类型及其大小等。

（6）添加图形注释，例如图名、坐标名称、图例、文字说明等。

（7）图形的导出与打印。

5.2　二维图形绘制

二维图形是 MATLAB 图形的基础，也是应用最广泛的图形类型之一。MATLAB 提供了许多二维图形绘制函数。

MATLAB 基本的二维图形包括线型（Line Plots）、散点型（Scatter Plots）、条型（Bar Plots）、极坐标型（Polar Plots）及矢量图型（Vector Fields）等。单击相应的类型，就可以查看相应的用法。

本节介绍二维图形的绘制函数 plot、图形的编辑、子图的绘制、双纵坐标图形的绘制及交互式图形的绘制。

5.2.1　plot 指令

将数对排序的一种方法是使用 plot 指令。该命令可以带有不同数目的参数。最简单的形式就是将数据传递给 plot，但是线条的类型和颜色可以通过使用字符串来指定，这里用 str 表示。线条的默认类型是实线型。

下面给出 plot 指令的一般使用规范。

（1）plot 指令使用规范一：plot(x,y)。

语句说明：以 x 为横坐标，y 为纵坐标，按照坐标(x_j,y_j)的有序排列绘制曲线。

（2）plot 指令使用规范二：plot(y)。

语句说明：其中 y 为一维实数数组，以 1:n 为横坐标，y_j 为纵坐标，绘制曲线（n 为 y 的长度）。

（3）plot 指令使用规范三：plot(z)。

语句说明：其中 z 为一维复数数组，以横轴为实轴，纵轴为虚轴，在复平面上绘制

(real(z_j),imag(z_j))的有序集合的图形。

例 5-4，plot 指令使用示例 1。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
X = -10:10; Y = X.^2;
figure(1)
subplot(1,3,1); plot(X, Y, 'LineWidth', 2);
subplot(1,3,2); plot(Y, 'LineWidth', 2);xlim([1 length(Y)])
Z = cos(-pi:pi/10:pi) + sqrt(-1)*sin(-pi:pi/10:pi);
subplot(1,3,3); plot(Z, 'LineWidth', 2);
```

运行 M 文件，结果如图 5-6 所示。

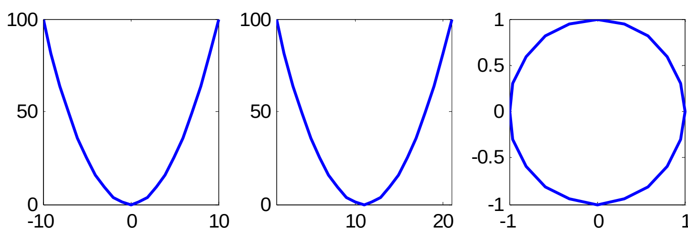


图 5-6　cosfigure.m 运行结果

（4）plot 指令使用规范四：plot(A)。

语句说明：绘制矩阵 ***A*** 的列对它的下标的图形。对于 $m \times n$ 的矩阵 ***A***，有 *n* 个含有 *m* 个元素的数对，或是 *n* 条有 *m* 个点的曲线，且这 *n* 条曲线均采用颜色监视器上不同的颜色绘制而成。

例 5-5，plot 指令使用示例 2。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
A = magic(20);
A(9:20,:) = [];
figure; plot(A)
```

运行 M 文件，结果如图 5-7 所示。

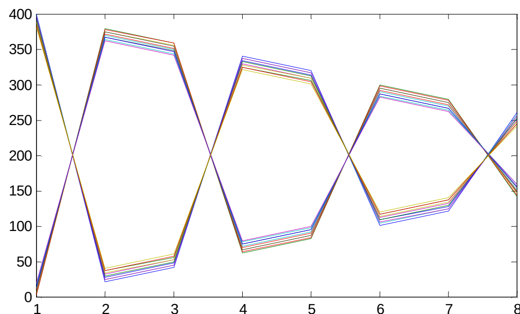


图 5-7　magicfigure.m 运行结果

（5）plot 指令使用规范五：plot(x,A)。

语句说明：绘制矩阵 ***A*** 对向量 ***x*** 的图形。对 $m \times n$ 的矩阵 ***A*** 和长度为 *m* 的向量 ***x***，绘制

矩阵 ***A*** 的列对向量 ***x*** 的图形；如果 ***x*** 的长度为 ***n***，则绘制矩阵 ***A*** 的行对向量 ***x*** 的图形。向量 ***x*** 可以是行向量，也可以是列向量。

（6）plot 指令使用规范六：plot(A,x)。

语句说明：对矩阵 ***A*** 绘制向量 ***x*** 的图形。对于一个 $m \times n$ 的矩阵 ***A*** 和一个长度为 ***m*** 的向量 ***x***，对矩阵 ***A*** 的列绘制向量 ***x*** 的图形；如果 ***x*** 的长度为 ***n***，则对矩阵 ***A*** 的行绘制向量 ***x*** 的图形。向量 ***x*** 可以是行向量，也可以是列向量。

（7）plot 指令使用规范七：plot(A,B)。

语句说明：对矩阵 ***A*** 的行绘制矩阵 ***B*** 的列的图形。如果 ***A*** 和 ***B*** 都是 $m \times n$ 的矩阵，则将绘制 ***n*** 条由 ***m*** 个有序对连成的曲线。

例 5-6，plot 指令使用示例 3。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
for i=1:5
   for j=1:6
      A(i,j)=i+j;
   end
end
x = 0.2:0.2:1;
figure(1)
subplot(2,2,1); plot(A,x,'LineWidth',1.5);
subplot(2,2,2); plot(x,A,'LineWidth',1.5);
B = reshape(1:30,5,6);
subplot(2,2,3); plot(A,B,'LineWidth',1.5);
subplot(2,2,4); plot(B,A,'LineWidth',1.5);
```

运行 M 文件，结果如图 5-8 所示。

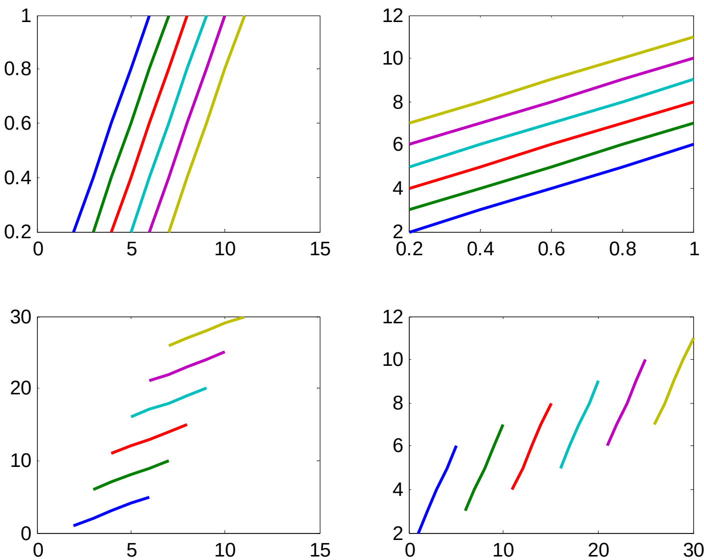


图 5-8　figuregrid.m 运行结果

（8）plot 指令使用规范八：plot(x,y,str)。

语句说明：用字符串 str 指定的颜色和线型对 ***y*** 绘制 ***x*** 的图形。

（9）plot 指令使用规范九：plot(x1,y1,str1, x2,y2,str2,⋯)。

语句说明：用字符串 str1 指定的颜色和线型对 *y*1 绘制 *x*1 的图形，用字符串 str2 指定的颜色和线型对 *y*2 绘制 *x*2 的图形……每组参数值可以采用上述除复数值以外的任何一种形式。str1,str2⋯可以省略，此时，MATLAB 自动为每条曲线选择颜色和线型。

例 5-7，plot 指令使用示例 4。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
j = sqrt(-1);
x = 2; y = 0.4;
bluecircle = cos(-pi:pi/20:pi) + j*sin(-pi:pi/20:pi) + (j*y - x);
blackcircle = cos(-pi:pi/20:pi) + j*sin(-pi:pi/20:pi) + (j*y);
redcircle = cos(-pi:pi/20:pi) + j*sin(-pi:pi/20:pi) + (j*y + x);
yellowcircle = cos(-pi:pi/20:pi) + j*sin(-pi:pi/20:pi) + (- j*y - x/2);
greencircle = cos(-pi:pi/20:pi) + j*sin(-pi:pi/20:pi) + (- j*y + x/2);
figure(1)
plot(bluecircle,'LineWidth',5);hold on;
plot(blackcircle,'k','LineWidth',5);hold on;
plot(redcircle,'r','LineWidth',5);hold on;
plot(yellowcircle,'y','LineWidth',5);hold on;
plot(greencircle,'g','LineWidth',5);
```

运行 M 文件，结果如图 5-9 所示。

例 5-8，plot 指令使用示例 5。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'--rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g',
'MarkerSize',10);
```

运行 M 文件，结果如图 5-10 所示。

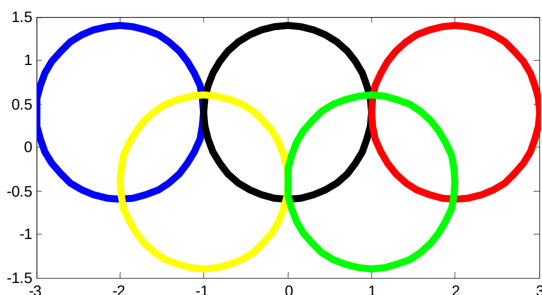


图 5-9　fivecircle.m 运行结果

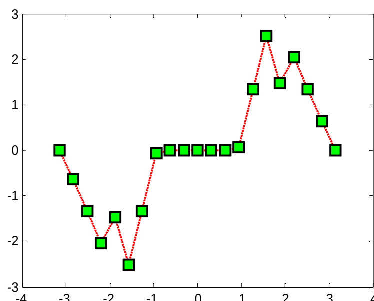


图 5-10　cosfigure.m 运行结果

例 5-9，利用函数 plot 绘制包络线。

在命令行窗口中输入如下语句：

```
clear all
t= (0:pi/100:pi)';          % <1>
y1=sin(t)*[1,-1];           % <2>
y2=sin(t).*sin(9*t);  % <3>
t3=pi*(0:9)/9;              % <4>
```

```
y3=sin(t3).*sin(9*t3);
plot(t,y1,'r:',t,y2,'b',t3,y3,'bo') % <5>
axis([0,pi,-1,1])          % <6>
```

输出结果如图 5-11 所示。

例 5-10，输入参数为向量时，利用函数 plot 绘制多条曲线。

在命令行窗口中输入如下语句：

```
clear all
x=-pi:pi/10:pi;
y=[sin(x);sin(x+3);sin(x+5)];
z=[cos(x);cos(x+3);cos(x+5)];
figure;
plot(x,y,'r:*',x,z,'g-.v');%绘制多条曲线
```

输出结果如图 5-12 所示。

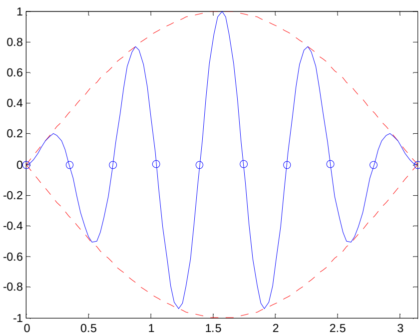


图 5-11　运行结果

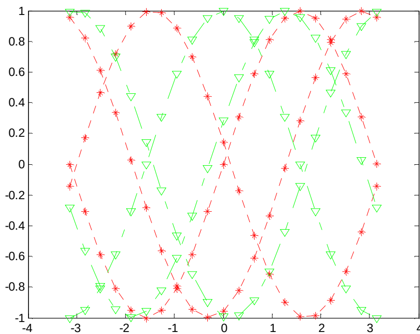


图 5-12　运行结果

5.2.2　格栅

当图像需要对具体数值有更加清楚的展示时，在图形中添加格栅是十分有效的方法。在 MATLAB 中，grid on 命令可以在当前图形的单位标记处添加格栅；grid off 命令则可以取消格栅的显示；单独使用 grid 命令则可以在 on 与 off 状态下交替转换，即起到触发的作用。

例 5-11，格栅的使用示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
X = (0:1800)*pi/180; Y = cos(X/2);
figure(1)
subplot(1,3,1); plot(X, Y, 'LineWidth', 2); xlim([0 30]); grid on;
subplot(1,3,2); plot(X, Y, 'LineWidth', 2); xlim([0 30]); grid on; grid;
subplot(1,3,3); plot(X, Y, 'LineWidth', 2); xlim([0 30]); grid; grid off;
```

运行 M 文件，结果如图 5-13 所示。

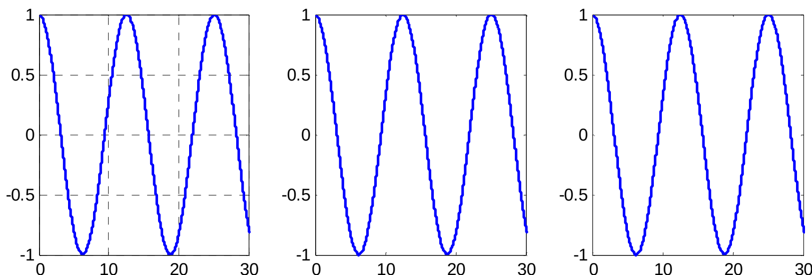


图 5-13　figuregrid.m 运行结果

5.2.3　文字说明

通常，曲线所表示的函数或数据的规律都需要进行一些文字说明或标注。图形窗口中的文本操作指令如下：

- title('text')：在图形窗口顶端的中间位置输出字符串“text”作为标题。
- xlabel('text')：在 *x* 轴下的中间位置输出字符串“text”作为标注。
- ylabel('text')：在 *y* 轴边上的中间位置输出字符串“text”作为标注。
- zlabel('text')：在 *z* 轴边上的中间位置输出字符串“text”作为标注。
- text(x,y, 'text')：在图形窗口的(*x*,*y*)处写字符串“text”。坐标 *x* 和 *y* 按照与所绘制图形相同的刻度给出。对于向量 *x* 和 *y*，字符串“text”写在(x_i,y_i)的位置上。如果“text”是一个字符串向量，即一个字符矩阵，且与 *x*、*y* 有相同的行数，则第 *i* 行的字符串将写在图形窗口的(x_i,y_i)的位置上。
- text(x,y, 'text', 'sc')：在图形窗口的(*x*,*y*)处输出字符“text”，给定左下角的坐标为(0.0,0.0)，右上角的坐标则为(1.0,1.0)。gtext('text')通过使用鼠标或方向键，移动图形窗口中的十字光标，让用户将字符串 text 放置在图形窗口中。当十字光标走到所期望的位置时，用户按下任意键或单击任意按钮，字符串将会写入窗口中。
- legend(str1,str2,...pos)：在当前图形上输出图例，并用说明性字符串 str1、str2 等做标注。其中，参数 pos 的可选项目如表 5-1 所示。

表 5-1　曲线线型

线型代号	表示线型
-1	将图例框放在坐标轴外的右侧
0	将图例框放在图窗内与曲线交叠最小的位置
1	将图例框放在图窗内右上角
2	将图例框放在图窗内左上角
3	将图例框放在图窗内左下角
4	将图例框放在图窗内右下角

- legend(str1,str2,...,'Location','pos')：在当前图形上输出图例，并用说明性字符串 str1、str2 等做标注。其中，参数 pos 的可选项目如表 5-2 所示。

表 5-2　图形标记

标记代号	表示标记	标记代号	表示标记
North	图窗内最上端	SouthOutside	图窗外下部
South	图窗内最下端	EastOutside	图窗外右侧
East	图窗内最右端	WestOutside	图窗外左侧
West	图窗内最左端	NorthEastOutside	图窗外右上部
NorthEast	图窗内右上角（二维图窗的默认项）	NorthWestOutside	图窗外左上部
NorthWest	图窗内左上角	SouthEastOutside	图窗外右下部
SouthEast	图窗内右下角	SouthWestOutside	图窗外左下部
SouthWest	图窗内左下角	Best	图窗内与曲线交叠最小的位置
NorthOutside	图窗外上部	BestOutside	图窗外最不占空间的位置

- legendoff：从当前图形中清除图例。

例 5-12，图窗内文字说明示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
x = 0:0.01*pi:pi*0.5; y = cos(x)+sqrt(-1)*sin(x);
plot(y*2,'r','LineWidth',5);hold on;
x = pi*0.5:0.01*pi:pi; y = cos(x)+sqrt(-1)*sin(x);
plot(y*2,'y','LineWidth',5);hold on;
x = -pi:0.01*pi:-pi*0.5; y = cos(x)+sqrt(-1)*sin(x);
plot(y*2,'b','LineWidth',5);hold on;
x = -pi*0.5:0.01*pi:0; y = cos(x)+sqrt(-1)*sin(x);
plot(y*2,'g','LineWidth',5);hold on;
title('极坐标系');
text([1.5 -3 1.5 -3],[2 2 -2 -2], {'第一象限','第二象限','第三象限','第四象限'})
legend({'[0  0.5\pi]','[0.5\pi  \pi]','[\pi  1.5\pi]','[1.5\pi  2\pi]'})
xlim([-5 5]);ylim([-5 5]);
plot([-4 4],[0 0],'k','LineWidth',3);hold on;
plot([0 0],[-4 4],'k','LineWidth',3);hold on;
axis off
```

运行 M 文件，结果如图 5-14 所示。

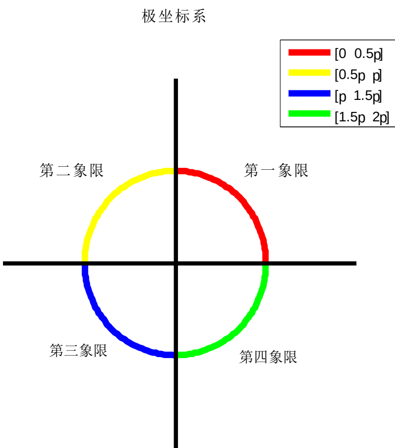


图 5-14　radioaxis.m 运行结果

MATLAB 中的字符串可以对输出的文字风格进行预先设置，可以预先规定的有字体、风格及大小。另外，可以进行上、下标的表示，还可以输出数学公式中经常用到的希腊字符和其他特殊字符。这些指令及特殊符号的输出方法如表 5-3～表 5-6 所示。

表 5-3　字体式样设置规则

字体指令：\fontname{arg}		文字风格\arg		文字大小\fontsize{arg}	
Arial	Arial 字体	bf	黑体	10	默认值，字号为小五
Roman	Roman 字体	it	斜体一	12	五号字体
宋体	字体为宋体	sl	斜体二		
黑体	字体为黑体	rm	正体		

表 5-4　上、下标的控制指令

	指令	arg 取值	示范	文字输出效果
上标	^{arg}	任何合法字符	'x^5+2x^4+4x^2+5x+6'	$x^5+2x^4+4x^2+5x+6$
下标	_{arg}		'x_1+x_2+x_3+x_4+x_5'	$x_1+x_2+x_3+x_4+x_5$

表 5-5　希腊字母

指令	字符	指令	字符	指令	字符	指令	字符
\alpha	a	\Theta	Q	\Pi	P	\psi	y
\beta	b	\iota	i	\rho	r	\Psi	Y
\gamma	g	\kappa	k	\sigma	s	\omega	w
\Gamma	G	\lambda	l	\Sigma	S	\Omega	W
\delta	d	\Lambda	L	\tau	t	\varpi	v
\Delta	D	\mu	m	\upsilon	u	\vartheta	J
\epsilon	e	\Nu	n	\Upsilon	U	\varsigma	V
\zeta	z	\xi	x	\phi	f		
\eta	h	\Xi	E	\Phi	F		
\theta	q	\pi	p	\chi	c		

表 5-6　其他特殊符号

指令	字符	指令	字符	指令	字符	指令	字符
\approx	»	\times	´	\perp	^	\Im	Â
\cong	@	\oplus	Å	\prime	'	\Re	Á
\div	¸	\oslash	Æ	\cdot	×	\downarrow	¯
\equiv	º	\otimes	Ä	\ldots	×××	\leftarrow	¬
\geq	³	\int	ò	\cap	Ç	\leftrightarrow	«
\leq	£	\partial	¶	\cup	È	\rightarrow	®
\neq	¹	\exists	$	\subset	Ì	\uparrow	-
\pm	±	\forall	"	\ subseteq	Í	\circ	°
\propto	µ	\in	Î	\supset	É	\bullet	·
\sim	~	\ infty	¥	\ supseteq	Ê	\copyright	Ó

5.2.4　线型、标记和颜色

当同一张图形中同时画了多条曲线时，需要使用不同的线型、标记或颜色来区分不同的曲线。

1．线型

MATLAB 平台中共有 5 种不同线型，如表 5-7 所示。

表 5-7　曲线线型

线型代号	表示线型
-	实线
--	虚线
-.	点画线
:	点线
none	无线

2．标记

MATLAB 平台中共有 14 种不同的标记方式，如表 5-8 所示。

表 5-8　图形标记

标记代号	表示标记	标记代号	表示标记
.	点	o	o
*	星号	+	+
square	正方形	x	×
diamond	菱形	<	顶点指向左边的三角形
pentagram	五角星形	>	顶点指向右边的三角形
hexagram	六角星形	^	正三角形
none	无点	v	倒三角形

3．颜色

MATLAB 平台中有代号的颜色共有 8 种，如表 5-9 所示。

表 5-9　曲线或标记颜色

颜色代号	表示颜色
g	绿色
m	品红色
b	蓝色
c	灰色
w	白色
r	红色
k	黑色
y	黄色

例 5-13，线型、标识与颜色示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
figure
x = 0:0.01*pi:pi*8;
plot(x,sin(x),'r:','LineWidth',3);hold on;
plot(x,2*sin(x/2),'y','LineWidth',3);hold on;
plot(x,4*sin(x/4),'b--','LineWidth',3);hold on;
x = 0:pi:pi*8;
plot(x,sin(x),'g^','MarkerSize',10,'LineWidth',3);hold on;
plot(x,2*sin(x/2),'co','MarkerSize',10,'LineWidth',3);hold on;
plot(x,4*sin(x/4),'msquare','MarkerSize',10,'LineWidth',3);hold on;
xlim([0 pi*8])
```

运行 M 文件，结果如图 5-15 所示。

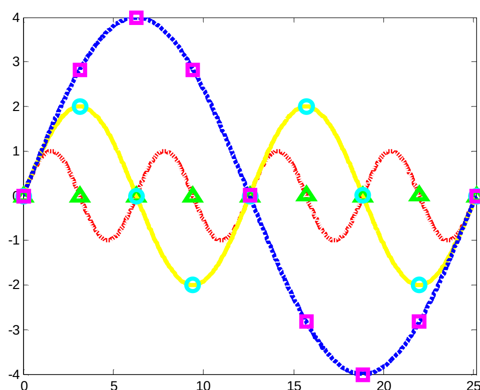


图 5-15　multicurve.m 运行结果

在 MATLAB 平台上还有一种叫作颜色映像的数据结构来代表颜色值。颜色映像定义为一个有 3 列和若干行的矩阵。利用 0～1 之间的数值，矩阵的每一行都代表了一种色彩。任一行的数字都指定了一个 RGB 值，即红、绿、蓝 3 种颜色的强度，形成一种特定的颜色。一些有代表性的 RGB 值如表 5-10 所示。

表 5-10　简单颜色

Red（红）	Green（绿）	Blue（蓝）	颜色
0	0	0	黑
1	1	1	白
1	0	0	红
0	1	0	绿
0	0	1	蓝
1	1	0	黄
1	0	1	洋红
0	1	1	青蓝
2/3	0	1	天蓝

（续表）

Red（红）	Green（绿）	Blue（蓝）	颜色
1	1/2	0	橘黄
.5	0	0	深红
.5	.5	.5	灰色

有 10 个 MATLAB 函数产生预定的颜色映像，如表 5-11 所示。

表 5-11　颜色映像函数集

函数名称	函数说明
Hsv	色彩饱和值（以红色开始和结束）
Hot	从黑到红到黄到白
Cool	青蓝和洋红的色度
Pink	粉红的彩色度
Gray	线性灰度
Bone	带一点蓝色的灰度
Jet	hsv 的一种变形（以蓝色开始和结束）
copper	线性铜色度
Prim	三棱镜。交替为红色、橘黄色、黄色、绿色和天蓝色
Flag	交替为红色、白色、蓝色和黑色

在默认情况下，上面所列的各个颜色映像产生一个 64×3 的矩阵，指定了 64 种颜色 RGB 的描述。这些函数都接收一个参量来指定所产生矩阵的行数。比如 hot(*m*)产生一个 *m*×3 的矩阵，它包含的 RGB 颜色值的范围从黑经过红、橘红、黄……一直到白。

大多数计算机在一个 8 位的硬件查色表中一次可以显示 256 种颜色，当然有些计算机的显卡可以同时显示更多的颜色。这就意味着在不同的图中，一般一次可以用 3 个或 4 个 64×3 的颜色映像。如果使用了更多的颜色映像输入项，则计算机必须经常在它的硬件查色表中调出输入项。例如，在画 MATLAB 图形时背景图案发生了变化，就是发生了这种情况。所以，除非计算机有一次显示更多种颜色的显卡，最好任何一次所用的颜色映像输入项数都小于 256。

5.2.5　坐标轴设置

图形坐标轴的取值范围及其刻度对图形的显示效果有着很明显的影响。在默认情况下，MATLAB 通过便捷智能的函数和内部自适应设置来显示图形。

有时候，默认设置生成的图形往往达不到用户所要求的效果，或者用户只对图形中的某一部分感兴趣时，就需要通过坐标轴控制函数，有针对性地调整和设置坐标轴的某些参数。

MATLAB 提供的坐标轴控制函数如表 5-12 所示。

表 5-12　MATLAB 的坐标轴控制函数

命　　令	描　　述	命　　令	描　　述
axis auto	使用坐标轴的默认设置	axis(xmin,xmax,ymin,ymax)	分别设定 x 轴、y 轴的坐标范围为[xmin,xmax]及[ymin,ymax]
axis manual	保持当前坐标刻度范围	axis equal	横、纵坐标采用等长刻度
axis fill	在 manual 方式下有效，使坐标充满整个绘图区	axis image	横、纵坐标采用等长刻度，且坐标框紧贴数据范围
axis off	取消坐标轴标签、刻度及背景	axis tight	把数据范围直接设定为坐标范围
axis on	打开坐标轴标签、刻度及背景	axis square	使用正方形坐标系
axis ij	使用矩阵式坐标，原点在左上方	axis normal	使用默认矩形坐标系，取消单位刻度的限制
axis xy	使用直角坐标，原点在左下方		

例 5-14，坐标轴设置示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
X = (0:1800)*pi/180; Y = cos(X/2);
figure(1)
subplot(1,3,1); plot(X, Y, 'LineWidth', 2); xlim([0 30]); grid on;
subplot(1,3,2); plot(X, Y, 'LineWidth', 2); xlim([0 30]); grid on; grid;
subplot(1,3,3); plot(X, Y, 'LineWidth', 2); xlim([0 30]); grid; grid off;
```

运行 M 文件，结果如图 5-16 所示。

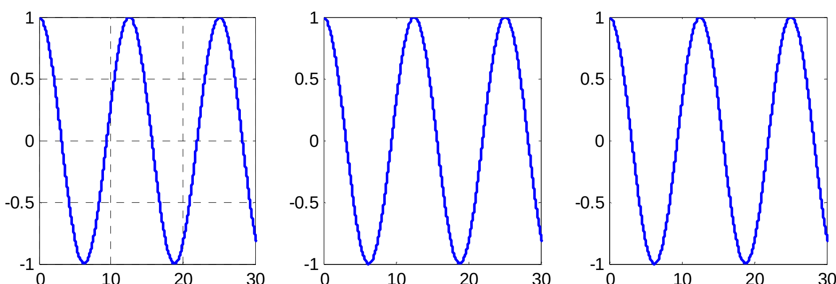


图 5-16　figureaxis.m 运行结果

5.2.6　图形叠绘

plot 指令可以在同一次调用中画出多条曲线，此功能已经在 5.2.1 节中进行了介绍。在实际应用中，经常遇到在已经存在的图上绘制新的曲线，并保留原来的曲线，MATLAB 中的以下指令可以完成这项功能。

- hold on 语句：语句功能为使当前轴及图形保留下来而不被刷新，并接收即将绘制的新的曲线。
- hold off 语句：语句功能为不保留当前轴及图形，绘制新的曲线后，原图即被刷新。
- hold 语句：语句功能为 hold on 语句与 hold off 语句的切换。

例 5-15，图形叠绘示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
figure
x = 0:0.01*pi:pi*4; y = 0:pi:pi*8;
subplot(1,2,1)
plot(x,sin(x),'r:','LineWidth',3);hold on;
plot(x,2*sin(x/2),'b','LineWidth',3);hold on;
plot(y,sin(y),'g^','MarkerSize',10,'LineWidth',3);hold on;
plot(y,2*sin(y/2),'mo','MarkerSize',10,'LineWidth',3);hold on;xlim([0 pi*4])
subplot(1,2,2)
plot(x,sin(x),'r:','LineWidth',3);
plot(x,2*sin(x/2),'b','LineWidth',3);hold on;
plot(y,sin(y),'g^','MarkerSize',10,'LineWidth',3);
plot(y,2*sin(y/2),'mo','MarkerSize',10,'LineWidth',3);hold on;xlim([0 pi*4])
```

运行 M 文件，结果如图 5-17 所示。

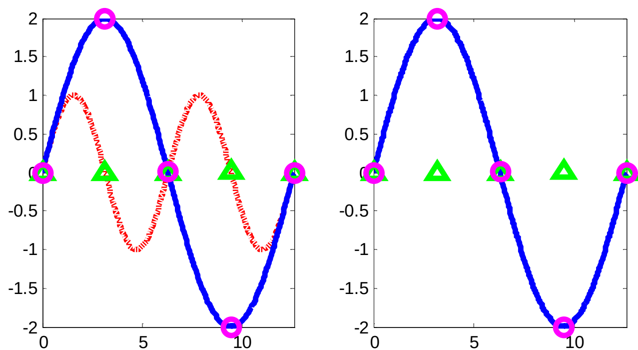


图 5-17　figurehold.m 运行结果

5.2.7　子图绘制

MATLAB 允许用户在同一个图形窗中同时绘制多幅相互独立的子图，这需要应用到 subplot 指令，其句法格式如下：

- subplot(m, n, k)：将（*m*´ *n*）幅子图中的第 *k* 幅图作为当前曲线的绘制图。
- subplot('position', [left bottom width height])：在指定位置上生成子图，并作为当前曲线的绘制图。

Subplot 指令说明：

（1）subplot(m, n, k)指令生成的图形窗中将会有（*m*´ *n*）幅子图，*k* 是子图的编号，编号的顺序为：左上为第一幅子图，然后向右后向下依次排号，该指令产生的子图分割与占位完全按照默认值自动进行。

（2）subplot('position', [left bottom width height])指令所产生的子图的位置由用户指定，指定位置的 4 个元素采用归一化的标称单位，即认为图形窗的宽、高的取值范围均为[0,1]，左下角的坐标为(0,0)。

（3）指令所产生的子图彼此之间相互独立，所有的绘图指令都可以在任一子图中运用，而对其他的子图不起作用。

（4）在使用 subplot 指令之后，如果再想绘制充满整个图形窗的图时，则应当先使用 clf 指令对图窗进行清空。

例 5-16，子图绘制示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
figure
x = 0:0.01*pi:pi*16; j = sqrt(-1);
subplot(2,2,1);plot(abs(sin(x)).*(cos(x)+j*sin(x)),'LineWidth',3);
xlim([-1 1]);ylim([-1 1]);
subplot(2,2,2);plot(abs(sin(x/2)).*(cos(x)+j*sin(x)),'LineWidth',3);
xlim([-1 1]);ylim([-1 1]);
subplot(2,2,3);plot(abs(sin(x/3)).*(cos(x)+j*sin(x)),'LineWidth',3);
xlim([-1 1]);ylim([-1 1]);
subplot(2,2,4);plot(abs(sin(x/4)).*(cos(x)+j*sin(x)),'LineWidth',3);
xlim([-1 1]);ylim([-1 1]);
```

运行 M 文件，结果如图 5-18 所示。

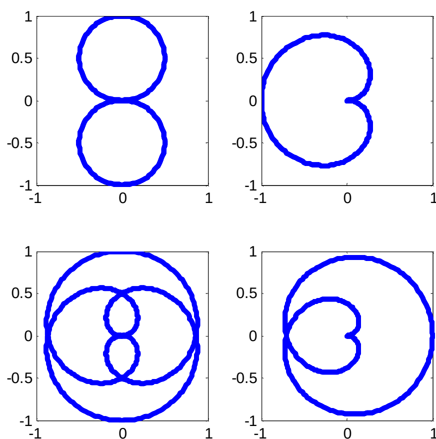


图 5-18　figuresub.m 运行结果

5.2.8　交互式绘图

MATLAB 中还设置了相应的鼠标操作的图形操作指令，分别是 ginput、gtext 和 zoom 函数。首先对这 3 个函数进行一般性的说明。

（1）除了 ginput 函数只能应用于二维图形，其余两个函数对二维和三维图像均适用。

（2）ginput 函数与 zoom 函数配合使用，可以从图形中获得较为准确的数据。

（3）在逻辑顺序并不十分清晰的情况下，并不提倡这几个指令同时使用。

1．ginput 指令

其调用格式如下：

- [x, y]=ginput(n)：其功能为用鼠标从二维图形中获得 *n* 个点的数据坐标(*x*,*y*)。

使用 ginput 指令时需要注意的是，指令中的 *n* 应当赋值为正整数，指令中的 *x* 和 *y* 用来存放所取点的坐标。该指令运行之后，会将当前图像从后台调度到前台。同时，鼠标光标变为十字形。用户可以移动鼠标，将其定位于待取点的位置处，单击则可获得该点的数据值，

然后通过相同的方式取得之后的 n-1 组数据值，当 n 组数据全部取得之后，图形窗便退回后台，回到 ginput 指令执行前的环境中。

2．gtext 指令

其调用格式如下：

- gtext(arg)：其功能为用鼠标把字符串或字符串元胞数组放置到图形中作为文字说明。

使用 gtext 指令时需要注意的是，运行该指令后，会将当前的图像从后台调度到前台。同时，鼠标光标变为十字形。用户可以移动鼠标，将其定位于待放置的位置处，右击，字符串将被放在紧靠十字中心点的“第一象限”位置上。

如果输入的 arg 是单个字符串，单击则可以一次性将所有字符以单行的形式放置在图形之中；如果 arg 中包含多行字符串，则每次单击可将其中的一行字符串放置在图形之中，直到将所有的字符串全部放置在图形之中后，操作才全部完成。

3．zoom 指令

zoom 指令的具体应用句法有几种格式，如表 5-13 所示。

表 5-13　缩放指令格式

指令格式	说　明
zoom xon	规定当前图形的 x 轴可以进行缩放
zoom yon	规定当前图形的 y 轴可以进行缩放
zoom on	规定当前图形可以进行缩放
zoom off	规定当前图形不可进行缩放
zoom	当前图形是否可以进行缩放状态的切换
zoom out	使图形返回初始状态
zoom(factor)	设置缩放变焦因子，默认值为 2

使用 zoom 指令时需要注意的是，变焦操作方式与标准的 Windows 缩放相同，在可变焦的当前图形上，可直接单击进行图形的放大，也可以长按鼠标左键括住需要放大的区域，放开鼠标左键之后，指定的区域则会被放大，右击则可进行图形的缩小。

默认的变焦因子是 2，即单击时图形被放大一倍，右击时图形被缩小一半。

5.2.9　双坐标轴绘制

在实际应用中，常常需要把同一自变量的两个不同量纲、不同量级的函数量的变化同时绘制在同一个图形窗中。例如，在同一张图中同时展示空间一点上的电磁波的幅度和相位随时间的变化；不同时间内的降雨量和温湿度的变化；放大器的输入/输出电流的变化曲线等。MATLAB 中的 plotyy 函数可以实现上述功能，其调用格式如下：

- plotyy(X1, Y1, X2, Y2)：以左、右不同的纵轴分别绘制 *X*1-*Y*1 和 *X*2-*Y*2 两条曲线。
- plotyy(X1, Y1, X2, Y2, Fun)：以左、右不同的纵轴以 Fun 指定的形式分别绘制

*X*1-*Y*1 和 *X*2-*Y*2 两条曲线。

- plotyy(X1, Y1, X2, Y2, Fun1, Fun2)：以左、右不同的纵轴分别以 Fun1、Fun2 指定的形式绘制 *X*1-*Y*1 和 *X*2-*Y*2 两条曲线。

使用 plotyy 指令时需要注意的是，左侧的纵轴用来描述 *X*1-*Y*1 曲线，右侧的纵轴用来描述 *X*2-*Y*2 曲线。轴的范围与刻度值都是自动生成的，进行人工设置时，使用的绘图指令与一般的绘图指令相同。

例 5-17，双坐标轴绘制示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
figure
angl = 0:0.01*pi:2*pi;
ampl = sin(0:0.01*pi:2*pi);
z = ampl.*(cos(angl) + sqrt(-1)*sin(angl));
[AX,H1,H2] = plotyy(0:200,abs(z),0:200,angle(z)*180/pi);
set(get(AX(1),'Ylabel'),'String','amplitude')
set(get(AX(2),'Ylabel'),'String','phase')
set(H1,'LineWidth',3)
set(H2,'LineStyle',':','LineWidth',3)
```

运行 M 文件，结果如图 5-19 所示。

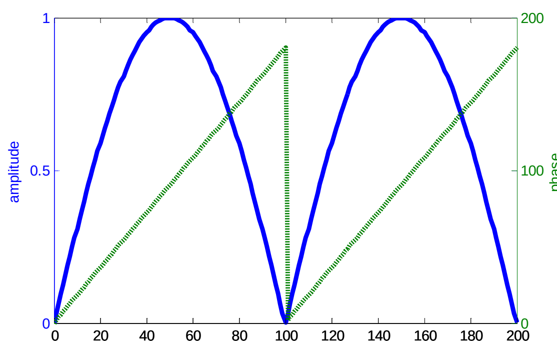


图 5-19　doubleaxis.m 运行结果

5.2.10　fplot 绘图指令

之前应用到的 plot 指令，均是将用户指定的或者计算得到的数据转换为图形。而在实际应用中，函数随着自变量的变化趋势是未知的，此时在 plot 指令下，如果自变量的离散间隔不合理，则无法反映函数的变化趋势。

fplot 指令可以很好地解决以上问题，该指令通过 MATLAB 平台内部设置的自适应算法来动态决定自变量的离散间隔，当函数值变化缓慢时，离散间隔取大一些；当函数值变化剧烈时，离散间隔取小一些。fplot 指令的调用格式如下：

- fplot(fun, limits)：在 limits 定义的自变量的取值范围[xmin,xmax]内，或者在自变量与因变量的取值范围[xmin,xmax;ymin,ymax]内，绘制 fun 函数。
- fplot(fun, limits, LineSpec)：在 limits 定义的取值范围内，在 LineSpec 所规定的线型、颜色、标记等属性下，绘制 fun 函数。

- fplot(fun, limits, tol)：在 limits 定义的取值范围内，在 tol 所规定的相对误差允许范围内，绘制 fun 函数。
- fplot(fun, limits, tol, LineSpec)：在 limits 定义的取值范围内，在 LineSpec 所规定的线型、颜色、标记等属性下，在 tol 所规定的相对误差允许范围内，绘制 fun 函数。
- fplot(fun, limits, n)：在 limits 定义的取值范围内绘制 fun 函数，至少绘制 n+1 个点。

使用 fplot 指令时需要注意的是：tol 为相对误差允许范围，其默认值为 2e-3，n 的默认值为 1，即 fplot 指令至少绘制两个点，对 n 进行设置后，最大的步长限制为(xmax-xmin)/n。

例 5-18，fplot 绘图示例。

在 MATLAB 命令行窗口中输入以下命令：

```
clear all
clc
fplot(@sin,[-2*pi 2*pi])
grid on
title('sin(x) from -2\pi to 2\pi')
xlabel('x');
ylabel('y');
```

运行 M 文件，结果如图 5-20 所示。

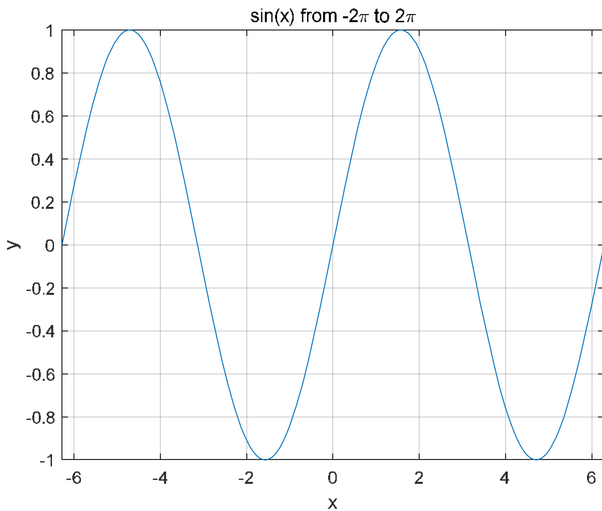


图 5-20　运行结果

5.2.11　ezplot 绘图指令

ezplot 指令用于绘制函数在某一自变量区域内的图形。与 fplot 指令相同的是，ezplot 指令中也需要对自变量的范围进行规定，其调用格式如下：

```
ezplot(f)
```

该语句的功能为：按 MATLAB 默认方式（自变量范围为$-2\pi < x < 2\pi$）绘制函数。

```
ezplot(f, [min, max])
```

该语句的功能为：设置 x 方向的变量范围为[min, max]，y 方向按 MATLAB 默认方式绘

制函数。

```
ezplot(f, [xmin,xmax, ymin, ymax])
```

该语句的功能为：x 方向的变量范围为[xmin,xmax]，y 方向的变量范围为[$y$min, $y$max]，绘制函数。

```
ezplot(x, y)
```

该语句的功能为：按 MATLAB 默认方式（自变量范围为$-2\pi < x < 2\pi$）绘制函数 x、y。

```
ezplot(x, y, [min, max])
```

该语句的功能为：按自变量范围为[min, max]绘制函数 x、y。

例 5-19，explot 绘图示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
ezplot('sin(x)^2+4*cos(y)^2=4',[-5 5 -1 1]);
axis square
```

运行 M 文件，结果如图 5-21 所示。

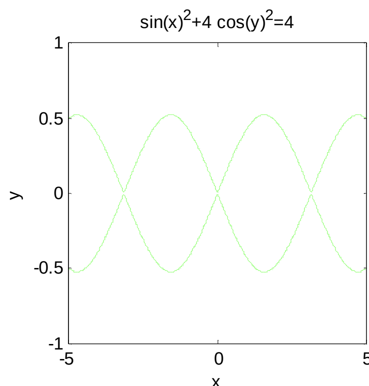


图 5-21　ellipse.m 运行结果

5.2.12　特殊坐标轴绘图

前几节介绍了基本的二维绘图函数的使用，但其中无论是直角坐标系还是极坐标系，用到的坐标轴的刻度均是线性刻度。但是在实际的很多情况中，数据都出现指数型的变化规律，这时如果再用线性刻度来描述曲线，则处于低次幂的部分数据无法清晰地表现出来。当然，利用 5.2.9 节中提到的双坐标轴绘图，对指数数据进行局部放大，则可以解决这个问题。但是在 MATLAB 中，还有更加简便的方式。本节介绍 3 个函数来解决对数数据的问题，分别是 semilogx 函数、semilogy 函数和 loglog 函数。

1．semilogx 函数

semilogx 函数的调用格式与 plot 基本相同，此处不再赘述每一个句法的功能。

- semilogx(Y)。
- semilogx(X1,Y1,X2,Y2,…)。
- semilogx(X1,Y1,LineSpec1,X2,Y2,LineSpec2,…)。

- semilogx(…, 'PropertyName ', PropertyValue,…)。

semilogx 函数绘制图形时，*x* 轴采用对数坐标。若没有指定使用的颜色，当所画线条较多时，semilogx 函数将自动使用由当前的 ColorOrder 和 LineStyleOrder 属性所指定的颜色顺序和线型顺序来绘制线条。

例 5-20，semilogx 函数与 plot 函数指令对比。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
x = 10.^(0.1:0.1:4);y = 1 ./ (x+1000);
figure
subplot(1,2,1);semilogx(x,y,'+','MarkerSize',5,'LineWidth',2);title('y =
 (x+1000)^{-1}')
subplot(1,2,2);plot(x,y,'+','MarkerSize',5,'LineWidth',2);title('y =
 (x+1000)^{-1}')
```

运行 M 文件，结果如图 5-22 所示。

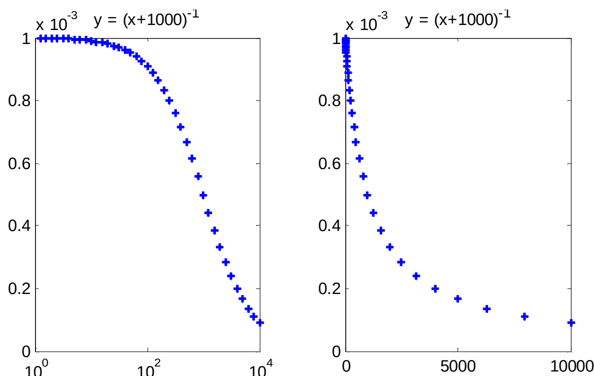


图 5-22　logx.m 运行结果

2．semilogy 函数

semilogy 函数的调用格式与 semilogx 相同，绘制图形时，*y* 轴采用对数坐标。若没有指定使用的颜色，当所画线条较多时，semilogy 将自动使用由当前的 ColorOrder 和 LineStyleOrder 属性所指定的颜色顺序和线型顺序来绘制线条。

3．loglog 函数

loglog 函数的调用格式与 semilogx 相同，绘制图形时，*x* 轴与 *y* 轴均采用对数坐标。

例 5-21，loglog 与 plot 指令对比。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
a = 0.1:0.1:5; x = log10(a); y = 10.^a;
figure
subplot(1,2,1);loglog(x,y,'+','MarkerSize',5,'LineWidth',2);title('lgy = 10^x')
subplot(1,2,2);plot(x,y,'+','MarkerSize',5,'LineWidth',2);title('lgy = 10^x')
```

运行 M 文件，结果如图 5-23 所示。

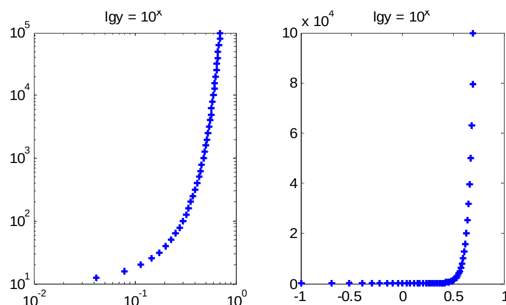


图 5-23 logxlogy.m 运行结果

5.2.13 二维特殊图形函数

在 MATLAB 中，除了可以通过 plot 函数等绘制特殊图形，还可以通过一些函数绘制柱状图、饼状图等特殊图形。常见的特殊二维图形函数，如表 5-14 所示。

表 5-14 二维特殊图形函数

函 数 名	说 明	函 数 名	说 明
area	填充绘图	fplot	函数绘制
bar	条形图	hist	柱状图
barh	水平条形图	pareto	Pareto 图
comet	彗星图	pie	饼状图
errorbar	误差带图	plotmatrix	分散矩阵绘制
ezplot	简单绘制函数图	ribbon	三维图的二维条状显示
ezpolar	简单绘制极坐标图	scatter	散射图
feather	矢量图	stem	离散序列火柴杆状图
fill	多边形填充	stairs	阶梯图
gplot	拓扑图	rose	极坐标系下的柱状图
compass	与 feather 功能类似的矢量图	quiver	向量场

表 5-14 中的函数均有不同的调用方法，下面通过示例向读者介绍其中几个常用的命令，其他函数的使用方法可以查阅 MATLAB 的在线帮助。

1．bar 函数

bar 函数用于绘制二维垂直条形图，用垂直条形显示向量或矩阵中的值。其调用格式如下：

- bar(y)：为每一个 *y* 中的元素画一个条状。
- bar(x,y)：在指定的横坐标 *x* 上画出 *y*，其中 *x* 为严格单增的向量。若 *y* 为矩阵，则 bar 把矩阵分解成几个行向量，在指定的横坐标处分别画出。
- bar(...,'bar_color ')：“bar_color”定义条的颜色。
- bar(axes_handle,...)：将图形绘制到坐标轴句柄 axes_handle 中，而不是当前坐标轴句柄中。

例 5-22，创建二维垂直条形图。

在命令行窗口中输入如下语句：

```
y = [75.995 91.972 105.711 123.203 131.669 ...
    150.697 179.323 203.212 226.505 249.633 281.422];
figure; bar(y);
```

运行结果如图 5-24 所示。

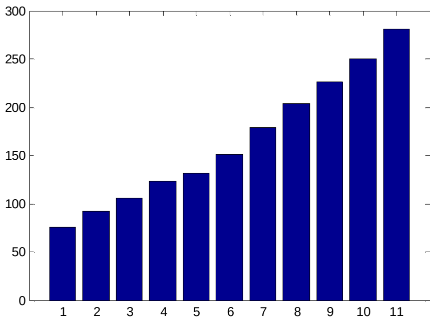


图 5-24　二维垂直条形图运行结果

例 5-23，二维直方图有两种类型：垂直直方图和水平直方图。而每种类型又有两种表现模式：累计式和分组式。本例选择其中两种加以表现。

在命令行窗口中输入如下语句：

```
x=-2:2;
Y=[6 8 7 4 5;4 8 1 12 0;4 6 21 1 3];
subplot(1,2,1),bar(x',Y','stacked')
xlabel('x'),ylabel('\Sigma y'),colormap(cool)
legend('因素 1','因素 2','因素 3')
subplot(1,2,2),barh(x',Y','grouped')
xlabel('y'),ylabel('x')
```

运行结果如图 5-25 所示。

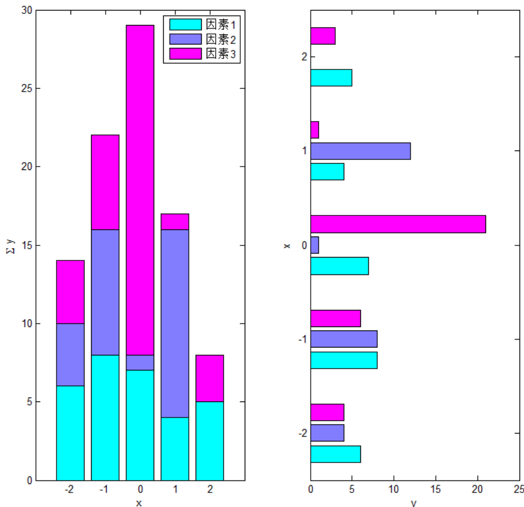


图 5-25　垂直直方图和水平直方图运行结果

2. pie 函数

pie 函数用于绘制饼形图。其调用格式如下：

- pie(x)：绘制参数 *x* 的饼图。
- pie(x, explode)：explode 是与 ***x*** 同维的矩阵，若其中有非零元素，***x*** 矩阵中相应位置的元素在饼图中对应的扇形将向外移出一些，加以突出。
- pie(...,labels)：labels 用于定义相应块的标签。
- pie(axes_handle,...)：将图形绘制到坐标轴句柄 axes_handle 中，而不是当前坐标轴句柄中。
- h = pie(...)：返回绘制的饼图相关的句柄。

例 5-24，创建二维饼图和三维饼图。

在命令行窗口中输入如下语句：

```
x = [1 5 0.5 3.5 2];
explode = [0 1 0 0 0];
pie(x,explode)%绘制饼图
colormap jet
%生成如图 5-26 所示的二维饼图
pie3(x,explode)
colormap hsv
```

运行结果如图 5-26 所示。

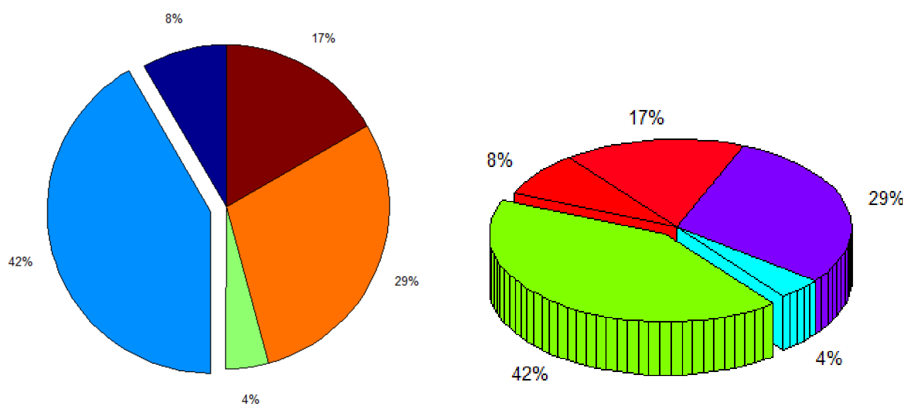


图 5-26　二维饼图和三维饼图程序运行结果

3. hist 函数

hist 函数用于绘制二维条形直方图，可以显示出数据的分布情况。所有向量 ***y*** 中的元素或者矩阵 ***Y*** 的列向量中的元素是根据它们的数值范围来分组的，每一组作为一个条形进行显示。条形直方图中的 *x* 轴反映了数据 *y* 中元素数值的范围，直方图的 *y* 轴显示出参量 *y* 中的元素落入该组的数目。其调用格式如下：

- n = hist(y)：把向量 ***y*** 中的元素放入等距的 10 个条形中，且返回每一个条形中的元素个数。若 ***y*** 为矩阵，则该命令按列对 ***y*** 进行处理。
- n = hist(y,x)：参量 ***x*** 为向量，把 ***y*** 中元素放到 *m*（*m*=length(*x*)）个由 ***x*** 中元素指定

的位置为中心的条形中。

- n = hist(y,nbins)：参量 nbins 为标量，用于指定条形的数目。
- [n,xout] = hist(...)：返回向量 ***n*** 与包含频率计数与条形的位置向量 xout，用户可以用命令 bar(xout,n)画出条形直方图。
- hist(...)：生成直方图，但不产生输出。
- hist(axes_handle,...)：将图形绘制到坐标轴句柄 axes_handle 中，而不是当前坐标轴句柄中。

例 5-25，绘制直方图。

在命令行窗口中输入如下语句：

```
x = -4:0.1:4;
y = randn(5000,1);
hist(y,x)     %绘制直方图
```

运行结果如图 5-27 所示。

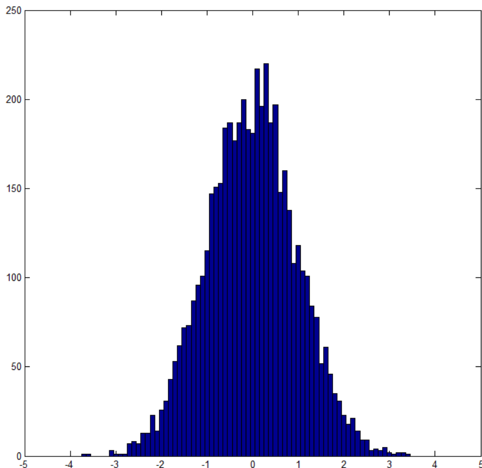


图 5-27　直方图程序运行结果

4．scatter 函数

scatter 函数用于绘制散点图。其调用格式如下：

- scatter(x,y)：以 *x*、*y* 的值为横、纵坐标，绘制散点。

例 5-26，绘制散点图示例。在命令行窗口中输入如下语句：

```
clear all
figure;
x=[1 5 6 7 9 5 1 3 12 20];
y=[20 15 6 3 1 5 3 0 1 5];
subplot(1,2,1);
scatter(x,y);                       %绘制散点图
subplot(1,2,2);
scatter(x,y,[],[1 0 0],'fill');    %绘制散点图
```

运行结果如图 5-28 所示。

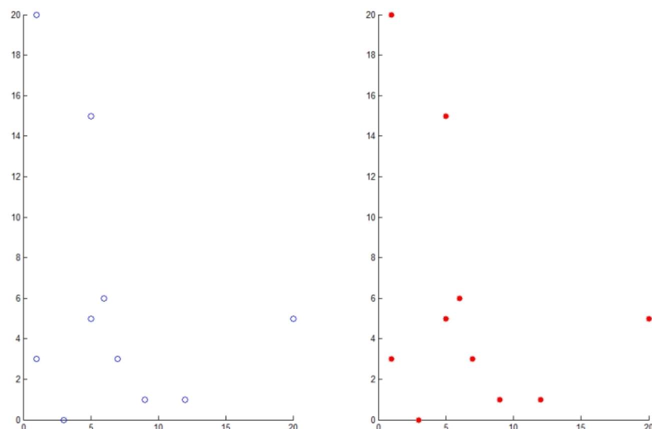


图 5-28　散点图程序运行结果

5.3　三维图形绘制

在实际的工程应用中，常常遇到三维甚至更多维的数据，需要在图形中表示出来，MATLAB 平台提供了相应的三维图形的绘制功能。这些绘图功能与二维图形的绘制十分类似，特别是曲线的属性，例如线型、颜色等的设置，是完全相同的。最常用的三维绘图有三维曲线图、三维网格图和三维曲面图这 3 种基本类型，相应的 MATLAB 指令分别为 plot3、mesh 和 surf。

5.3.1　曲线图绘制

在已经学习了 plot 指令的基础上，在三维图形指令中，plot3 指令十分易于理解，其调用格式也与 plot 指令类似，具体如下：

- plot3(X,Y,Z)：***X***、***Y***、***Z*** 为同维向量时，绘制以 ***X***、***Y***、***Z*** 为 *x*、*y*、*z* 坐标的三维曲线；***X***、***Y***、***Z*** 为同维矩阵时，用 ***X***、***Y***、***Z*** 的对应列元素绘制 *x*、*y*、*z* 坐标的三维曲线，曲线的条数为矩阵的列数。
- plot3(X1,Y1,Z1,X2,Y2,Z2)：绘制以 *X*1、*Y*1、*Z*1 和 *X*2、*Y*2、*Z*2 为 *x*、*y*、*z* 坐标的三维曲线。
- plot3(X,Y,Z, 'PropertyName', PropertyValue,…)：在 PropertyName 所规定的曲线属性下，绘制以 *X*、*Y*、*Z* 为 *x*、*y*、*z* 坐标的三维曲线。
- plot3(X1, Y1, Z1, 'PropertyName1', PropertyValue1, X2, Y2, Z2, 'PropertyName2', PropertyValue2)：在 PropertyName1 所规定的曲线属性下，绘制以 *X*1、*Y*1、*Z*1 为 *x*、*y*、*z* 坐标的三维曲线；在 PropertyName2 所规定的曲线属性下，绘制以 *X*2、*Y*2、*Z*2 为 *x*、*y*、*z* 坐标的三维曲线。需要说明的是，plot3 指令用来表现的是单参数的三维曲线，而非双参数的三维曲面。

例 5-27，plot3 指令使用示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
theta = 0:.01*pi:2*pi; x = sin(theta); y = cos(theta); z = cos(4*theta);
figure
plot3(x,y,z,'LineWidth',2);hold on;
theta = 0:.02*pi:2*pi; x = sin(theta); y = cos(theta); z = cos(4*theta);
plot3(x,y,z,'rd','MarkerSize',10,'LineWidth',2)
```

运行 M 文件，结果如图 5-29 所示。

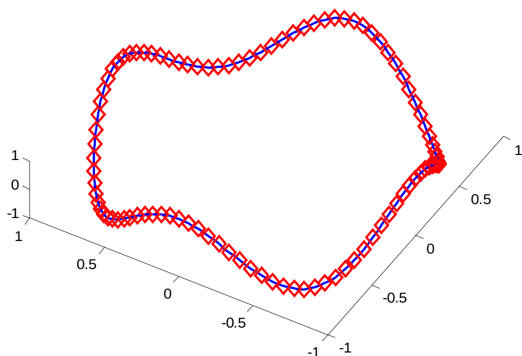


图 5-29　三维曲线运行结果

5.3.2　网格图绘制

三维网格图和曲面图的绘制比三维曲线图的绘制稍显复杂，主要是因为绘图数据的准备及三维图形的色彩、明暗、光照和视角等的处理。绘制函数 $z=f(x,y)$的三维网格图的过程如下：

（1）确定自变量 x 和 y 的取值范围和取值间隔如下：

```
x=x1:dx:x2, y=y1:dy:y2
```

（2）构成 xoy 平面上的自变量采样“格点”矩阵。

①利用“格点”矩阵的原理生成矩阵。

```
x=x1:dx:x2; y=y1:dy:y2;
X=ones(size(y))*x;
Y=y*ones(size(x));
```

②利用 meshgrid 指令生成“格点”矩阵。

```
x=x1:dx:x2; y=y1:dy:y2;
[X,Y]=meshgrid(x,y);
```

（3）计算在自变量采样“格点”上的函数值：$Z=f(X,Y)$。

绘制网格图的基本 mesh 指令的调用格式如下：

- mesh(X,Y,Z)：以 X 为 x 轴自变量、Y 为 y 轴自变量，绘制网格图；$\boldsymbol{X}$、$\boldsymbol{Y}$ 均为向量，若 X、Y 长度分别为 m、n，则 $\boldsymbol{Z}$ 为 m´ n 的矩阵，即[m,n]=size($\boldsymbol{Z}$)，则网格线的顶点为(X_j,Y_i,Z_{ij})。
- mesh(Z)：以 $\boldsymbol{Z}$ 矩阵列下标为 x 轴自变量、行下标为 y 轴自变量，绘制网格图。

- mesh(X,Y,Z,C)：以 *X* 为 *x* 轴自变量、*Y* 为 *y* 轴自变量，绘制网格图；其中 *C* 用于定义颜色，如果不定义 *C*，则成为 mesh(*X*,*Y*,*Z*)，其绘制的网格图的颜色随着 *Z* 值的变化（即曲面高度）而变化。
- mesh(X,Y,Z, 'PropertyName ',PropertyValue,…)：以 *X* 为 *x* 轴自变量、*Y* 为 *y* 轴自变量，绘制网格图；PropertyValue 用来定义网格图的标记等属性。

例 5-28，mesh 指令使用示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
X = -10:0.1:10; Y = -10:0.1:10;
[X,Y] = meshgrid(X,Y);
Z = - X.^2 - Y.^2 + 200;
mesh(X,Y,Z)
```

运行 M 文件，结果如图 5-30 所示。

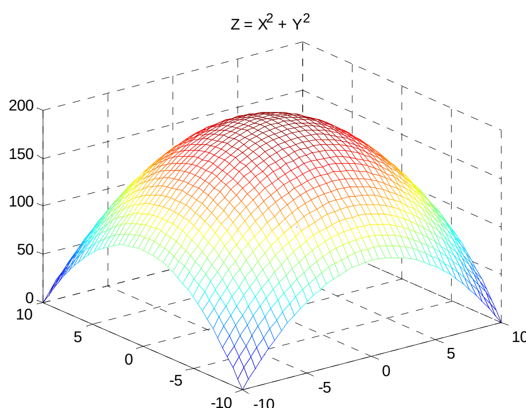


图 5-30　3dmesh.m 文件运行结果

5.3.3　曲面图绘制

曲面图的绘制由 surf 指令完成，该指令的调用格式与 mesh 指令类似，具体如下：

- surf (X,Y,Z)。
- surf (Z)。
- surf (X,Y,Z,C)。
- surf(X,Y,Z, 'PropertyName',PropertyValue,…)。

mesh 指令所绘制的图形是网格划分的曲面图，而 surf 指令绘制得到的是平滑着色的三维曲面图，着色的方式是在得到相应的网格点后，对每一个网格依据该网格所代表的节点的色值（由变量 *C* 控制）来定义这一网格的颜色。

例 5-29，surf 指令与 mesh 指令对比。

创建 M 文件并命名为 ex5_29.m。

利用 M 文件编辑器，在 M 文件中写入：

```
[x,y,z]=peaks(25);
figure
surf(x,y,z)
```

```
mesh(x,y,z)
```

运行 M 文件，结果如图 5-31 所示。

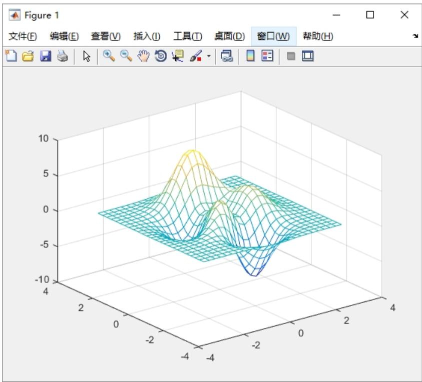


图 5-31　运行结果

5.3.4　光照模型

光照是一种利用方向光源照亮物体的技术。在某些情况下，这项技术能使表面微妙的差异更容易看到。光照也能用来对三维的图像增加现实感。

例 5-30，带光照的曲面图。

创建 M 文件并命名为 ex5_30.m。

利用 M 文件编辑器，在 ex5_30.m 文件中写入：

```
figure
surf(X,Y,Z,'FaceColor','red','EdgeColor','none');
camlight left; lighting phong
view(-15,65)
```

运行 M 文件，结果如图 5-32 所示。

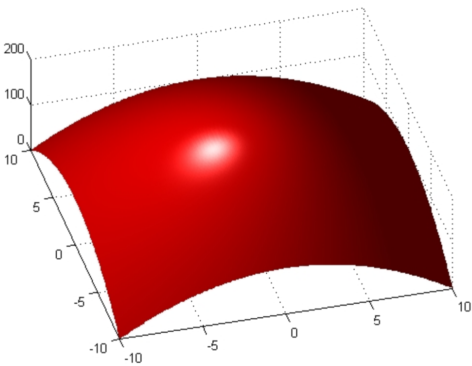


图 5-32　光照的曲面

本例中将曲面涂上了红色，并且将 surf 指令所定义的网格线移除。同时，一个发光的物体被加到了“镜头”的左边（即从空间观看时所在表面的位置）。增加光源和设置好照明方

式到 phong 后，使用 view 命令去改变视角，从空间的另一个不同的点再观看表面（方位角-15°和仰角 65°）。最后，用工具栏缩放方式放大外观。

基于运用漫射、镜面反光和环境照明模型，MATLAB 中还内置了 surfl 函数，可以画出类似于函数 surf 产生的带彩色的曲面。使用一个单色颜色映像（如灰色、纯白、铜黄或粉红色）和插值色彩，会画出效果更好的曲面。其调用格式如下：

- surfl(X,Y,Z,S)：其中 S 以[Sx,Sy,Sz]或[az,el]的形式定义光源方向。在没有明确定义的情况下，其默认光源是逆时针 45°。

例 5-31，surfl 指令应用示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
    [X,Y,Z]=peaks(30);
    subplot(1,2,1);surfl(X,Y,Z) , colormap(copper) , title('Default Lighting') , shading
interp
    subplot(1,2,2);surfl(X,Y,Z,[-90  30],[.55  .6  2  10]) , shading interp
```

运行 M 文件，结果如图 5-33 所示。

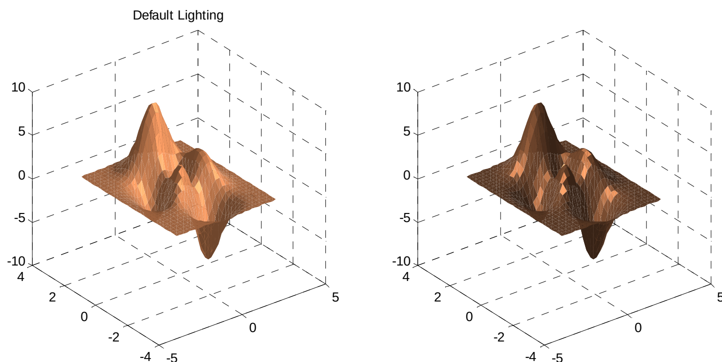


图 5-33　surfl 函数运行结果

值得注意的是，插值上色会极大地降低打印速度。这是因为每一像素都有一个不同的颜色值，打印机对每点都要分别上色。

5.3.5　绘制等值线图

等值线图又叫等高线图。绘制等值线图需要用到 contour 指令，其调用格式如下：

- contour(Z)：以 ***Z*** 矩阵列下标为 ***x*** 轴自变量、行下标为 ***y*** 轴自变量，绘制等值线图。
- contour(Z,n)：其中 ***n*** 为所绘制的图形等值线的条数。
- contour(Z,v)：其中 ***v*** 为向量，向量的长度为等值线的条数，并且等值线的值为对应的向量的元素值。
- contour(X,Y,Z)：以 ***X*** 为 ***x*** 轴自变量、***Y*** 为 ***y*** 轴自变量，绘制等值线图；***X***、***Y*** 均为向量，若 ***X***、***Y*** 长度分别为 ***m***、***n***，则 ***Z*** 为 ***m´ n*** 的矩阵，即[*m*,*n*]=size(*Z*)，则网格线的顶点为(X_j,Y_i,Z_{ij})。
- contour(X,Y,Z,n)：其中 ***n*** 为所绘制的图形等值线的条数。

- contour(X,Y,Z,v)：其中 **v** 为向量，向量的长度为等值线的条数，并且等值线的值为对应的向量的元素值。
- surf(…,LineSpec)：其中 LineSpec 用来定义等值线的线型。

与 contour 作用相类似的指令还有 contourf 指令，其调用格式与 contour 相同。

例 5-32，contour 指令使用示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
[X,Y,Z] = peaks(30);
figure
subplot(2,2,1);contour(X,Y,Z);axis square
subplot(2,2,2);contour(X,Y,Z,10);axis square
subplot(2,2,3);contour(X,Y,Z,-10:1:10);axis square
subplot(2,2,4);contour(X,Y,Z,':');axis square
```

运行 M 文件，结果如图 5-34 所示。

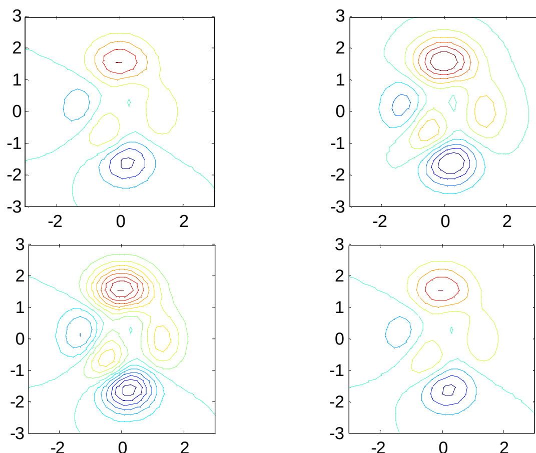


图 5-34　contour 函数运行结果

5.4　四维图形可视化

5.4.1　用颜色描述第四维

用 mesh 和 surf 等指令所绘制的图像，在未给出颜色参量的情况下，图像的颜色是沿着 *z* 轴的数据变化的。例如，surf(X,Y,Z)与 surf(X,Y,Z,Z)两个指令的执行效果是相同的。将颜色施加于 *z* 轴能够产生色彩亮丽的图画，但由于 *z* 轴已经存在，因此它并不提供新的信息。因此为了更好地利用颜色，可以考虑使用颜色来描述不受 3 个轴影响的数据的某些属性。为此，需要赋给三维作图函数的颜色参量所需要的“第四维”的数据。

如果作图函数的颜色参量是一个向量或矩阵，那么就用作颜色映像的下标。这个参量可以是任何实向量或与其参量维数相同的矩阵。

例 5-33，使用颜色描述第四维示例。

输入如下命令：

```
[X,Y,Z]=peaks(30);
R = sqrt(X.^2+Y.^2);
subplot(1,2,1);surf(X,Y,Z,Z);
axis tight
subplot(1,2,2);surf(X,Y,Z,R);
axis tight
```

程序运行结果如图 5-35 所示。

其中，在坐标系中描述一个面需要三维数据，而另一维数据描述空间中的点的坐标值，则使用不同的颜色表现出来；在左图中，第四维数据为 *Z*；在右图中，第四维数据为 *R*；在图上可以看到两者的颜色分布发生了明显的变化。

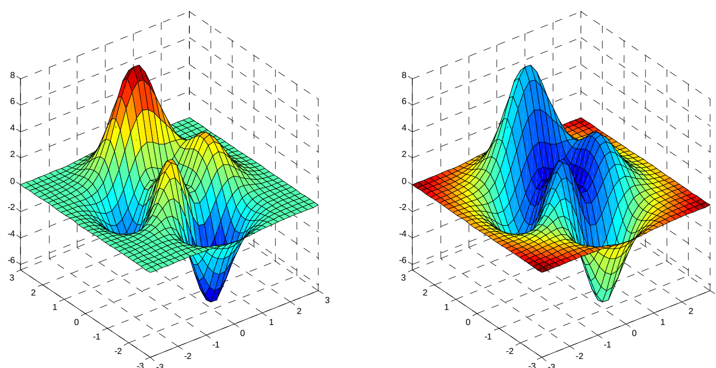


图 5-35　使用颜色描述第四维示例

5.4.2　其他函数

除了 surf 函数，mesh 和 pcolor 函数也可以将第四维的数据附加到颜色属性上，并在图像中表现出来。各函数的句法列表如表 5-15 所示。

表 5-15　其他指令的句法和功能

调用格式	说明
surf(X,Y,Z,fun(X,Y,Z))	根据函数 fun(*X,Y,Z*)来附加颜色数据
surf(X,Y,Z)=surf(X,Y,Z,Z)	默认动作，附加颜色数据于 *z* 轴
surf(X,Y,Z,X)	附加颜色数据于 *x* 轴
surf(X,Y,Z,Y)	附加颜色数据于 *y* 轴
surf(X,Y,Z,X.^2+Y.^2)	*xoy* 平面上距原点一定的距离附加颜色数据
surf(X,Y,Z,del2(Z))	根据曲面的拉氏函数值附加颜色数据
[dZdx,dZdy]=gradient(Z);surf(X,Y,Z,abs(dZdx))	根据 *x* 轴方向的曲面斜率附加颜色数据
dz=sqrt(dZdx.^2+dZdy.^2);surf(X,Y,Z,dz)	根据曲面斜率大小附加颜色数据

除了表 5-15 所列出的函数，slice 函数也可以通过颜色来表示存在于第四维空间中的值，其调用格式如下：

- slice(V, nx, ny, nz)：显示三元函数 *V*(*X*, *Y*, *Z*)确定的立体图在 *x* 轴、*y* 轴、*z* 轴方向上

的若干点（对应若干平面）的切片图，各点的坐标由数量向量 *sx*、*sy*、*sz* 指定。其中 *V* 为大小为 $m\times n\times p$ 的三维数组，默认值为 *X*=1:*m*、*Y*=1:*n*、*Z*=1:*p*。

- slice(X, Y, Z, V, nx, ny, nz)：显示三元函数 *V*(*X*, *Y*, *Z*)确定的立体图在 *x* 轴、*y* 轴、*z* 轴方向上的若干点（对应若干平面）的切片图。若函数 *V*(*X*, *Y*, *Z*)中有一个变量 *X* 取定值 X_0，则函数 $V(X_0, Y, Z)$为 $X=X_0$ 立体面的切面图（将该切面通过颜色表示 *V* 的值），各点的坐标由数量向量 *sx*、*sy*、*sz* 指定。参量 *X*、*Y*、*Z* 均为三维数组，用于指定立方体 *V* 的每点的三维坐标。
- slice(V, XI, YI, ZI)：显示由参量矩阵 ***XI***、***YI***、***ZI*** 确定的立体图的切片图，参量 ***XI***、***YI***、***ZI*** 定义了一个曲面，同时会在曲面的点上计算立体图 *V* 的值。需要注意的是，***XI***、***YI***、***ZI*** 必须为同型矩阵。
- slice(X, Y, Z, V, XI, YI, ZI)：沿着由矩阵 ***XI***、***YI***、***ZI*** 定义的曲面穿过立体图 *V* 的切片图。
- slice(…, 'method')：通过 method 来指定内插值的方法，method 可取 linear、cubic、nearest。linear 指定的内插值方法为三次线性内插值（若未指定，此即为默认值），cubic 指定使用三次立方内插值法，nearest 指定使用最近点内插值法。

5.5　本章小结

本章系统地阐述了将离散数据表示成可视化图形的基本过程，曲线、曲面绘制的基本技法和指令，以及特殊图形的生成和使用示例。读者在绘图时，应当根据实际情况选择合适的函数和算法，使得可视化图形的表示最合理，信息的表现最全面。例如可以使用不同线型、不同颜色、不同数据点标记来凸显不同数据的特征，还可以使用着色、灯光照明、反射效果、不同材质和透明度来处理渲染和烘托表现出高维函数的形状等。

第 2 部分

第 6 章

数据分析



知识要点

数据分析和处理是各种应用中非常重要的问题。针对数据分析和处理，MATLAB 提供了大量的函数，方便用户使用。本章将介绍 MATLAB 强大的数据分析和处理功能，首先介绍多项式函数，这些函数用于多项式求值、多项式乘法、多项式除法等；接着介绍插值函数，MATLAB 提供数个不同的插值算法；最后介绍 limit 函数对数组和函数极限的求解过程。

 学习要求	学习目标 / 知识点	了解	理解	应用	实践
	数据分析的基本概念		√		
	多项式及其函数			√	√
	数据插值方法				√
	数组与函数极限			√	√
	函数的数值积分				√

6.1 多项式及其函数

多项式作为线性方程组的表现形式，在运算及应用中具有非常重要的意义，本节将重点介绍多项式的各种运算法则、运算函数及操作指令，并特别对有理多项式进行说明。

6.1.1 多项式的表达式和创建

MATLAB 中使用一维向量来表示多项式，将多项式的系数按照降幂次序存放在向量中。多项式 $P(x)$的具体表示方法如下：

$P(x)= a_0x^n +a_1x^{n-1} +a_{n-1}x +a_n$的系数构成的向量为：[a_0 a_1… $a_{n-1}a_n$]。

例如，多项式 $2x^4 +3x^3 + 5x^2 + 1$ 就可以用向量[2 3 5 0 1]来表示。应当注意的是，多项式中缺少的幂次的系数应当为“0”。在 MATLAB 中，多项式由一个行向量表示，它的系数是按降序排列的。

例 6-1，输入多项式 $3x^4$-$10x^3$+$15x$+1000。

在命令行窗口中输入：

```
p=[3  -10  0  15  1000]
```

输出结果：

```
p =
3   -10    0    15   1000
```

○ 注意

必须包括具有 0 系数的项，例如上例中，多项式并没有二次项，因此二次项的系数为 0，否则 MATLAB 是无法知道哪一项为 0 的。

例 6-2，将向量表示的多项式用字符串输出的通用函数示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
function s=pprintf (p)
%pprintf: for polynomial printf.
%pprintf(p) printf p as string
%2012.01.23
if nargin>1
    error('Too much input arguments ')     %确认输入变量
end
while p(1)==0                              %将高次的零去掉
   p(1)=[];
end
l=length(p);                       %计算 p 的长度
if l==0
   s='0';
elseif l==1
   s=num2str(p(1));
elseif l==2
   s=strcat(num2str(p(1)),'x+',num2str(p(2)));
elseif l>2
```

```
    for i=1:l
       if i==1
          if p(i)==1
             s=strcat('x^{',num2str(l-i),'}');
          else
             s=strcat(num2str(p(i)),'x^{',num2str(l-i),'}');
          end
       elseif i==l
          s=strcat(s,'+',num2str(p(i))');
       elseif i==l-1
          s=strcat(s,'+',num2str(p(i)),'x');
       else
          if p(i)==0
          else
             s=strcat(s,'+',num2str(p(i)),'x^{',num2str(l-i),'}');
          end
       end
    end
end
```

现在，为了试验函数 pprintf，在命令行窗口中输入：

```
p=[1 0 3 5 7 9];
figure;title(pprintf(p))
```

输出结果：

$x^5+3x^3+5x^2+7x+9$

6.1.2　多项式求根

1．多项式的根

找出多项式的根，即使多项式为 0 的值，可能是许多学科共同的问题。MATLAB 能求解这个问题，并提供了特定函数 roots 求解一个多项式的根。

例 6-3，求解多项式 $3x^4-10x^3+15x+1000$ 的根。

在命令行窗口中输入：

```
p=[3  -10  0  15  1000];
r=roots(p)
```

输出结果：

```
r =
   4.0283 + 2.8554i
   4.0283 - 2.8554i
  -2.3616 + 2.8452i
  -2.3616 - 2.8452i
```

2．由根创建多项式

在 MATLAB 中，无论是一个多项式还是它的根，都是以向量形式存储的，按照惯例，多项式是行向量，根是列向量。因此当我们给出一个多项式的根时，MATLAB 也可以构造出相应的多项式，这个过程需要使用函数 poly。

例 6-4，由根创建多项式。

在命令行窗口中输入：

```
r =[4.0283 + 2.8554i; 4.0283 - 2.8554i; -2.3616 + 2.8452i; -2.3616 - 2.8452i];
p=poly(r)
```

输出结果：

```
p =
    1.0000   -3.3333    0.0000    5.0000  333.3333
```

因为 MATLAB 无隙地处理复数，当用根重组多项式时，如果一些根有虚部，由于截断误差，则 poly 的结果有一些小的虚部。消除虚假的虚部，只要使用函数 real 抽取实部即可。

6.1.3 多项式的四则运算

1．多项式的加法

对于多项式的加法，MATLAB 并未提供一个特别的函数。如果两个多项式向量大小相同，那么多项式相加时就与标准的数组加法相同。

例 6-5，多项式加法运算示例。

在命令行窗口中输入：

```
a=[1 3 5 7 9];
b=[1 2 4 6 8];
c=a+b
```

输出结果：

```
c =
    2    5    9   13   17
```

结果是 $c(x)= 2x^4+5x^3+9x^2+13x+17$。

> 注意
>
> 当两个多项式阶次不同时，低阶的多项式用首零填补，使其与高阶多项式有同样的阶次。要求首零而不是尾零，是因为相关的系数像 x 幂次一样，必须整齐。

例 6-6，多项式加法运算示例。

在命令行窗口中输入：

```
a=[1 3 5 7 9];
b=[0 2 4 6 8];
c=a+b
```

输出结果：

```
c =
1    5    9   13   17
```

结果是 $c(x)= x^4+5x^3+9x^2+13x+17$。

有的时候，两个相加的多项式的项数并不明确，这时可用一个文件编辑器创建一个通用的多项式相加函数 M 文件来执行一般的多项式加法。

例 6-7，通用的多项式加法函数示例。

创建 M 文件，利用 M 文件编辑器，在 M 文件中写入：

```
function p=ppadd(a,b)
%  ppadd: for polynomial addition.
```

```
% ppadd(a,b) adds the polynomial a and b
% 2012.01.23
if nargin<2
    error(' Not enough input arguments ')  %确认输入变量
end
a=a(:).' ;                                 %将输入转化为行向量
b=b(:).' ;
na=length(a) ;                             %求相加的两个多项式的项数
nb=length(b) ;
p=[zeros(1,nb-na) a]+[zeros(1,na-nb) b] ;  %相加的两个多项式项数不等时，高次补零
```

现在，为了试验函数 mmpadd，在命令行窗口中输入：

```
a=[1 3 5 7 9];
b=[0 2 4 6 8];
c=a+b
```

输出结果：

```
c =
1    5    9   13   17
```

结果是 $c(x)= x^4+5x^3+9x^2+13x+17$。

2．多项式的乘法

在 MATLAB 中，函数 conv 支持多项式乘法（运算法则为执行两个数组的卷积）。

例 6-8，多项式乘法运算示例。

在命令行窗口中输入：

```
a=[1 3 5 7 9];
b=[1 2 4 6 8];
c=conv(a , b)
```

输出结果：

```
c =
1    5   15   35   69   100   118   110   72
```

结果是 $c(x)=x^8+5x^7+15x^6+35x^5+69x^4+100x^3+118x^2+110x+72$。

3．多项式的除法

在一些特殊情况下，一个多项式需要除以另一个多项式。在 MATLAB 中，这是由函数 deconv 完成的。

例 6-9，多项式除法运算示例。

在命令行窗口中输入：

```
c=[1 5 15 35 69 100 118 110 72];
b=[1 2 4 6 8];
[a, r]=deconv (c , b)
```

输出结果：

```
a=
1    35    7    9
r=
0    0    0    0    0    0    0    0    0
```

a 是多项式 c 除以多项式 b 的商，余式为 r。本例中 r 为零多项式，因为多项式 b 和多项

式 a 的乘积恰好是 c。

6.1.4 多项式的导数、积分与估值

1. 多项式的导数

MATLAB 为多项式求导提供了函数 polyder。

例 6-10，多项式求导运算示例。

在命令行窗口中输入：

```
d=[1 5 15 35 69 100 118 110 72];
e=polyder(d)
```

输出结果：

```
e=
8    35    90   175   276   300   236   110
```

2. 多项式的积分

MATLAB 为多项式的积分提供了函数 polyint。其具体的调用格式如下：

- polyint(P, k)：返回多项式 *P* 的积分，积分常数项为 *k*。
- polyint(P)：返回多项式 *P* 的积分，积分常数项默认值为 0。

例 6-11，多项式积分运算示例。

在命令行窗口中输入：

```
d=[1 5 15 35 69 100 118 110 72];
f=polyint (d)
```

输出结果：

```
f =
  Columns 1 through 8
    0.1111    0.6250    2.1429    5.8333   13.8000   25.0000   39.3333   55.0000
  Columns 9 through 10
   72.0000         0
```

3. 多项式的估值

根据多项式系数的行向量，可对多项式进行加、减、乘、除和求导运算，也能对它们进行估值。在 MATLAB 中，这由函数 polyval 来完成。

例 6-12，多项式的估值。

在命令行窗口中输入：

```
x= -1:0.01:1;      %生成自变量
g=[1 3 5 7 9];     %函数对应的多项式
h=polyval(g,x);    %进行估值运算
plot(x , h);xlabel('x');title('x^4+3x^3+5x^2+7x+9');     %将估值运算结果对自变量作图
```

输出结果如图 6-1 所示。

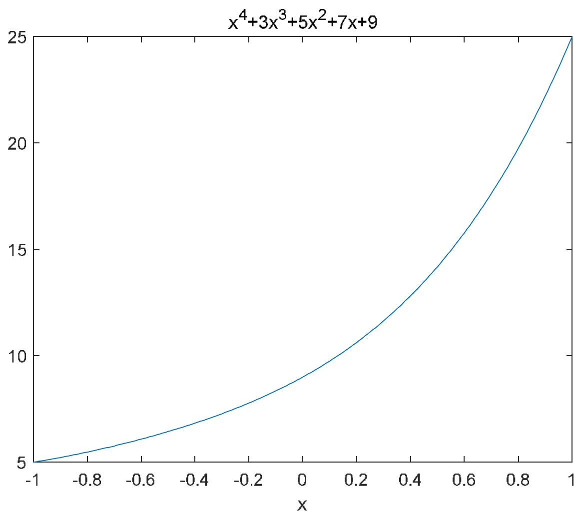


图 6-1　多项式估值运行结果

6.1.5　多项式运算函数及操作指令

多项式的运算函数及常用的操作指令如表 6-1 和表 6-2 所示。

表 6-1　多项式运算函数

多项式运算函数	函数作用
conv(*a*, *b*)	乘法
[q, r]=deconv(*a*, *b*)	除法
poly(*r*)	用根构造多项式
polyder(*a*)	对多项式或有理多项式求导
polyfit(*x*, *y*, *n*)	多项式数据拟合
polyval(*p*, *x*)	计算 *x* 点中多项式的值
[*r*, *p*, *k*]=residue(*a*, *b*)	部分分式展开式
[*a*, *b*]=residue(*r*, *p*, *k*)	部分分式组合
roots(*a*)	求多项式的根

表 6-2　多项式操作指令

多项式操作指令	指令含义
mmp2str(*a*)	多项式向量到字符串变换，a(*s*)
mmp2str(*a*, ' *x* ')	多项式向量到字符串变换，a(*x*)
mmp2str(*a*, ' *x* ', 1)	常数和符号多项式变换
mmpadd(*a*, *b*)	多项式加法
mmpsim(*a*)	多项式简化

6.1.6　有理多项式

在许多应用中，例如傅里叶（Fourier）、拉普拉斯（Laplace）和 Z 变换中，出现了两个多项式之比。在 MATLAB 中，有理多项式由它们的分子多项式和分母多项式表示。对有理

多项式进行运算的两个函数是 residue 和 polyder。函数 residue 执行部分分式展开的运算。

例 6-13，有理多项式的展开。

在命令行窗口中输入：

```
num=[ 5 3 -2 7];    %分子多项式
den=[-4 0 8 3];     %分母多项式
[r, p, k] = residue(num,den)
```

输出结果：

```
r =
   -1.4167
   -0.6653
    1.3320
p =
    1.5737
   -1.1644
   -0.4093
k =
   -1.2500
```

本例结果可表示成：

$$\frac{5x^3+3x^2-2x+7}{-4x^3+8x+3}=\frac{-1.4167}{x-1.5737}+\frac{-0.6653}{x+1.1644}+\frac{1.332}{x+0.4093}-1.25$$

residue 函数也可执行逆运算。

例 6-14，有理多项式展开的逆运算。

在命令行窗口中输入：

```
num=[ 5 3 -2 7];      %分子多项式
den=[-4 0 8 3];       %分母多项式
[r, p, k] = residue(num,den);
[n, d] = residue(r, p, k)
```

输出结果：

```
n =
   -1.2500   -0.7500    0.5000   -1.7500
d =
    1.0000   -0.0000   -2.0000   -0.7500
```

本例结果可表示成：

$$\frac{n(x)}{d(x)}=\frac{-1.25x^3-0.75x^2+0.5x-1.75}{x^3-2x-0.75}$$

函数 polyder 可用来对多项式求导。除此之外，如果输入两个多项式，则 polyder 指令对两个多项式构成的有理多项式求导。

例 6-15，有理多项式的求导。

在命令行窗口中输入：

```
num=[ 5 3 -2 7];      %分子多项式
den=[-4 0 8 3];       %分母多项式
[b, a]=polyder(num , den)
```

输出结果：

```
    b =
       12    64   153    18   -62
    a =
       16     0   -64   -24    64    48     9
```

6.2　数据插值

插值法是一种古老的数学方法。插值问题的数学定义如下：

由实验或测量的方法得到函数 $y=f(x)$ 在互异点 $x_0,x_1,\cdots,x_n$ 处的数值 $y_0,y_1,\cdots,y_n$（如图 6-2 所示），然后构造一个函数 $\varphi(x)$ 作为 $y=f(x)$ 的近似表达式，即

$$y=f(x)\approx\varphi(x)$$

使得 $\varphi(x_0)=y_0,\varphi(x_1)=y_1,\cdots,\varphi(x_n)=y_n$。这类问题称为插值问题。$y=f(x)$ 称为被插值函数，$\varphi(x)$ 称为插值函数，$x_0,x_1,\cdots,x_n$ 称为插值节点。

插值的任务是由已知的观测点为物理量建立一个简单的、连续的解析模型，以便能根据该模型推测该物理量在非观测点处的特性。

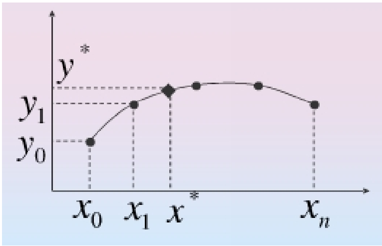


图 6-2　一维插值示意图

插值包括多项式插值、艾尔米特插值、分段插值与样条插值、三角函数插值、辛克插值等几种。

插值在数据分析、信号处理、图像处理等诸多领域有着十分重要的应用。

6.2.1　一维插值

当被插值函数 $y=f(x)$ 为一元函数时，为一维插值。MATLAB 使用 interp1 函数来实现一维插值。interp1 函数的调用格式如下：

- Vq = interp1(X,V,Xq,METHOD)：*X* 为自变量的取值范围；*V* 为函数值，或者 *V* 为一向量，其长度必须与 *X* 保持一致；*X*q 为插值点向量或数组；METHOD 是字符串变量，用来设定插值方法。

MATLAB 提供以下几种插值方法：

- METHOD='nearest'：邻近点插值。插值点函数值估计为与该插值点最近的数据点函数值。

- METHOD='linear'：线性插值法。根据相邻数据点的线性函数估计落在该区域内插值数据点的函数值。
- METHOD='spline'：三次样条插值。这种方法在相邻数据点间建立三次多项式函数，根据多项式函数确定插值数据点的函数值。
- METHOD='pchip'或'cubic'：立方插值。通过分段立方 Hermite 插值方法计算插值结果。
- METHOD='v5cubic'：用 MATLAB5 版本中断三次样条插值。

上述几种插值方法的比较如下：

- 邻近点插值方法的速度最快，但平滑性最差。
- 线性插值占用的计算机内存比邻近点插值多，运算时间也长；与邻近点插值不同，其结果是连续的，但顶点处的斜率会改变。
- 三次样条插值方法的运算时间最长，但内存的占用较立方插值少，其插值数据和导数都是连续的。在这几种插值中，三次样条插值结果的平滑性最好，但如果输入数据不一致或数据点过近，就可能出现很差的插值效果。

选择一种插值方法，需要考虑的因素包括运算时间、占用计算机内存的大小和插值的光滑程度。

下面介绍几种较为常用的一维插值方法。

1. 分段线性插值（linear）

分段线性插值的算法是在每个小区间$[x_i, x_{i+1}]$上采用简单的线性插值。在区间$[x_i, x_{i+1}]$上的子插值多项式为：

$$F_i = \frac{x - x_{i+1}}{x_i - x_{i+1}} f(x_i) + \frac{x - x_i}{x_{i+1} - x_i} f(x_{i+1})$$

在此整个区间$[x_i, x_n]$上的插值函数为：

$$F(x) = \sum_{i=1}^{n} F_i l_i(x)$$

其中 $l_i(x)$的定义如下：

$$l_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}} & x \in [x_{i-1}, x_i] \quad (i = 0) \\ \dfrac{x - x_{i+1}}{x_i - x_{i+1}} & x \in [x_i, x_{i+1}] \quad (i = 0) \\ 0 & x \notin [x_{i-1}, x_{i+1}] \end{cases}$$

例 6-16，采用 interp1 对 y=sin(x)进行分段线性插值。

在命令行窗口中输入如下语句：

```
clear all;
x=0:2*pi;
```

```
y=sin(x)
xx=0:0.5:2*pi;
yy=interp1(x,y,xx);
plot(x,y,'s',xx,yy)
```

得到的结果如图 6-3 所示。

2．一维快速傅里叶插值

一维快速傅里叶插值通过函数 interpft()来实现。该函数用傅里叶变换把输入数据变换到频域，然后用更多点的傅里叶逆变换变回时域，其结果是对数据进行增采样。函数 interpft()的调用格式如下：

- y = interpft(x,n)：对 *x* 进行傅里叶变换，然后采用 ***n*** 点傅里叶逆变换变回到时域。如果 ***x*** 是一个向量，数据 ***x*** 的长度为 ***m***，采样间隔为 d*x*，则数据 *y* 的采样间隔是 d*xm*/*n*。

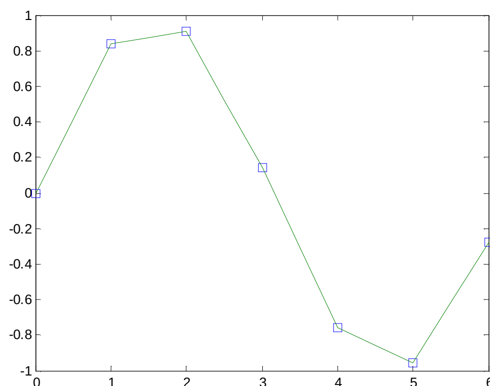


图 6-3　分段线性插值结果

- y = interpft(x,n,dim)：在 dim 指定的维度上进行操作。

例 6-17，利用一维快速傅里叶插值实现数据增采样。

在命令行窗口中输入如下语句：

```
%一维快速傅里叶插值实现数据增采样
clear all;
x = 0:1.2:10;
y = sin(x);
n = 2*length(x);      %增采样 1 倍
yi = interpft(y,n);   %一维快速傅里叶插值
xi = 0:0.6:10.4;
hold on;
plot(x,y,'ro');       %画图
plot(xi,yi,'b.-');
title('一维快速傅里叶插值');
legend('原始数据','插值结果');
```

得到的结果如图 6-4 所示。

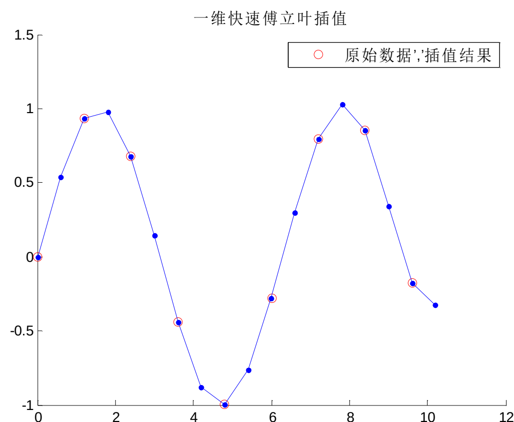


图 6-4　一维快速傅里叶插值结果

3．快速 fourier 算法

当数据点呈现周期分布时，用上面的几种插值算法效果都不是很好，这时可以使用 interpft 函数进行插值，此函数使用快速 fourier 算法作一维插值，其调用格式如下：

```
y = interpft(x,n)
```

注意

它返回周期函数在重采样的 *n* 个等距点的插值。注意，*n* 必须大于 *x* 的长度。

例 6-18，采用 interpft 函数对 sin 函数插值。

在命令行窗口中输入如下语句：

```
clear all;
x=0:2*pi;
y=sin(x)
z=interpft(y,15);
xx=linspace(0,2*pi,15);%生成 0～2pi 之间的 15 个线性等分点
plot(x,y,'-o',xx,z,':o')
```

得到的结果如图 6-5 所示。

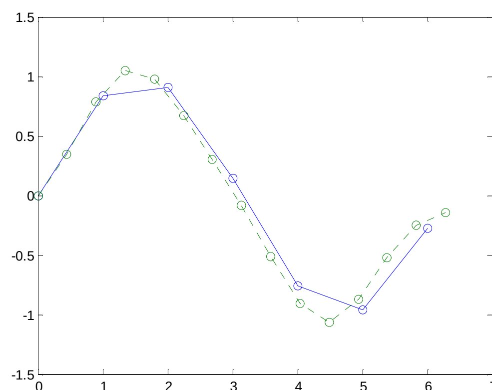


图 6-5　快速 fourier 插值结果

6.2.2　二维插值

当被插值函数 $y=f(x)$ 为二元函数时，为二维插值。MATLAB 使用 interp2 函数来实现二维插值。interp2 函数的调用格式如下：

- Vq = interp2(X,Y,V,Xq,Yq,METHOD)：***X***、***Y***、***V*** 是具有相同大小的矩阵，*V*(*i*,*j*)是数据点[*X*(*i*,*j*),*Y*(*i*,*j*)]上的函数值；*X*q、*Y*q 为待插值数据网格；METHOD 是一个字符串变量，表示不同的插值方法。

MATLAB 提供以下 4 种插值方法：

- METHOD='nearest'：领近点插值。将插值点周围 4 个数据点中离该插值点最近的数据点函数值作为该插值点的函数值的估计值。
- METHOD='linear'：双线性插值，是 MATLAB 的 interp2 函数默认使用的插值方法。该方法将插值点周围 4 个数据点的函数值的线性组合作为插值点的函数值的估计值。
- METHOD='spline'：三次样条插值。该方法的计算效率高，得到的曲面光滑，是用户经常使用的插值方法。
- METHOD='cubic'：双立方插值。该方法利用插值点周围的 16 个数据点，相对于 nearest 和 linear 方法，需要消耗较多的内存和计算时间，故计算效率不高，但是得到的曲面更加光滑。

例 6-19，二维插值示例。

在命令行窗口中输入如下语句：

```
clear all;
[X,Y] = meshgrid(-3:.25:3);         %产生已知的数据栅格点
Z = peaks(X,Y);                            %计算已知点的函数值
[XI,YI] = meshgrid(-3:.125:3);             %产生更精密的插值点
ZI = interp2(X,Y,Z,XI,YI);
mesh(X,Y,Z), hold, mesh(XI,YI,ZI+15)
hold off
axis([-3 3 -3 3 -5 20])
```

输出结果如图 6-6 所示。

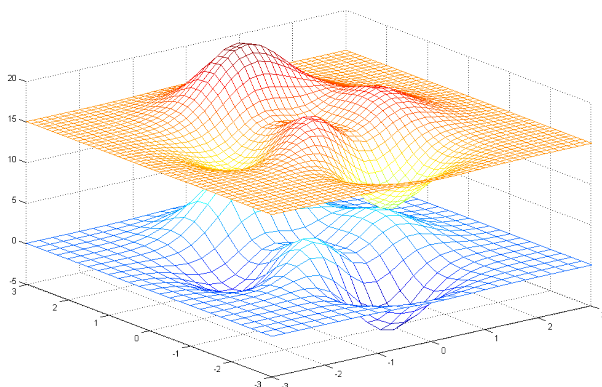


图 6-6　函数二维插值结果

6.3 函数的极限

极限理论是微积分学的基础理论。在 MATLAB 中，采用 limit 计算数量或函数的极限。

6.3.1 极限的概念

设 $\{x_n\}$ 为数列，a 为常数。若对任给的正数 ε，总存在正整数 N，使得当 $n>N$ 时有 $|x_n-a|<\varepsilon$，则称数列$\{x_n\}$收敛于 a，常数 a 称为数列 $\{x_n\}$ 的极限，并记作

$$\lim_{n\to\infty} x_n = a$$

6.3.2 求极限的函数

当 $x\to x_0^-$时，若函数 f(x)以 a 为极限，则称为函数 f(x)当 $x\to x_0^-$时，以 a 为左极限；当 $x\to x_0^+$时，若函数 f(x)以 a 为极限，则称为函数 f(x)当 $x\to x_0^+$时，以 a 为右极限。左极限和右极限统称单侧极限，当左极限和右极限同时存在且相等时，称 $\lim_{n\to x_0^+} f(x)$ 存在且等于 a。

在 MATLAB 中采用 limit 函数求某个具体函数的极限，其常用的调用格式如下：

- limit(expr,x,a)：当 $x\to a$ 时，对函数 expr 求极限，返回值为函数极限。
- limit(expr)：默认当 $x\to 0$ 时，对函数 expr 求极限，返回值为函数极限。
- limit(expr,x,a,'left')：当 $x\to a$ 时，对函数 expr 求其左极限，返回值为函数极限。
- limit(expr,x,a,'right')：当 $x\to a$ 时，对函数 expr 求其右极限，返回值为函数极限。

例 6-20，对于数列$\{n/(3n+1)\}$，当 n 趋于无穷大时，求其极限。

在命令行窗口中输入如下语句：

```
clear all
n=1:200;
y=n./(3*n+1);
figure;
plot(n,y); %显示数列
syms x;
f=x/(3*x+1);
z=limit(f,x,inf)
```

输出结果如下（如图 6-7 所示）：

```
z =
 1/3
```

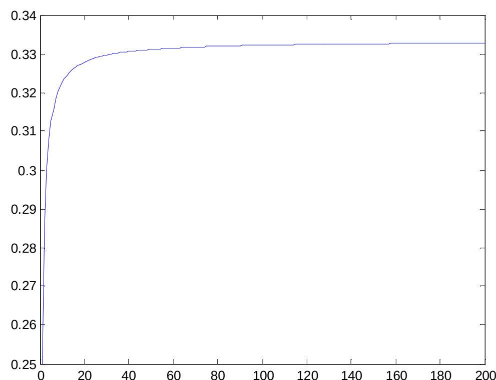


图 6-7　数列的极限

例 6-21，求极限 $\lim_{x\to 1}\dfrac{3x^2}{3x^2-2x+1}$。

在命令行窗口中输入如下语句：

```
clear all
syms x;
f=(3*x^2)/(3*x^2-2*x+1);
z=limit(f,x,1)
```

输出结果：

```
z =
 3/2
```

例 6-22，求极限 $\lim_{x\to 0}\dfrac{\sin(\sin(x))}{x}-1$。

在命令行窗口中输入如下语句：

```
clear all
syms x;
f=sin(sin(x))/x-1;
z=limit(f,x,0)
```

输出结果：

```
z =
0
```

6.4　函数数值积分

定积分的计算可用牛顿-莱布尼茨公式：

$$\int_a^b f(x)\mathrm{d}x = F(b) - F(a)$$

其中，$F(x)$是 $f(x)$的原函数之一，可用不定积分求得。然而在实际问题中，在应用上述公式时往往遇到一系列的问题。

- 被积函数 $f(x)$是使用函数表格提供的。
- 被积函数表达式极为复杂，求不出原函数；或者求出的原函数的形式很复杂，不利于计算。
- 大量函数的原函数不容易或者根本无法求出，例如正弦型积分 $\int_0^1 \frac{\sin x}{x}\mathrm{d}x$ 等，根本无法用初等函数来表示其原函数，因而无法精确计算其定积分。

数值积分便是为了解决上述问题而提出来的。数值积分只需计算 $f(x)$在节点 $x_i(i=1,2,\cdots,n)$上的值，计算方便且适合在计算机上实现。

6.4.1 数值积分问题的数学表述

区间$[a,b]$上的定积分$\int_a^b f(x)\mathrm{d}x$，就是在区间$[a,b]$内取$n+1$个点$x_0,x_1,\cdots,x_n$，利用被积函数$f(x)$在这$n+1$个点的函数值的某一种线性组合来近似作为待求定积分的值，即

$$\int_a^b f(x)\mathrm{d}x \approx \sum_{k=0}^{n} A_k f(x_k)$$

其中，x_k称为积分节点，A_k称为求积系数。右端公式称为左端定积分的某个数值积分公式。因此，求积分的关键在于积分节点x_k的选取及积分系数A_k的确定。

MATLAB 支持三重以下的积分运算，分别为：$\int_{x_{\min}}^{x_{\max}} f(x)\mathrm{d}x$、$\int_{x_{\min}}^{x_{\max}}\int_{y_{\min}}^{y_{\max}} f(x,y)\mathrm{d}x\mathrm{d}y$、$\int_{x_{\min}}^{x_{\max}}\int_{y_{\min}}^{y_{\max}}\int_{z_{\min}}^{z_{\max}} f(x,y,z)\mathrm{d}x\mathrm{d}y\mathrm{d}z$。在计算积分值时，要求积分区间是确定的。

6.4.2 一元函数的数值积分

本节将讨论$f(x)$为一元函数时的积分情况。MATLAB 为一元函数的数值积分提供了 3 个函数，分别为 quad、quadl、quadv。下面将对这 3 个函数进行介绍。

1. quad 函数

quad 采用遍历的自适应辛普森（Simpson）法计算函数的数值积分，适用于精度要求低、被积函数平滑性较差的数值积分。常用的调用格式如下：

- Q = quad(FUN,A,B)。
- Q = quad(FUN,A,B,TOL)。

- Q = quad(FUN,A,B,TOL,TRACE)。
- [Q,FCNT] = quad(...)。

式中，FUN 为被积函数的句柄，FUN 应该接收向量输入，并输出相同长度的向量；*A*、*B* 分别为积分的起始值和结束值；TOL 用于控制自适应辛普森法的误差，增大 TOL 可以加快计算速度，但是计算精度下降。默认情况下，TOL=1.0e-6；TRACE 值非 0 时，函数输出计算过程中的[fcnt a b-a Q]；FCNT 表示函数计算的次数。

2．quadl 函数

quadl 函数采用遍历的自适应 Lobatto 法计算函数的数值积分，适用于精度要求高、被积函数曲线比较平滑的数值积分。其用法与 quad 相同，分别如下：

- Q = quadl(FUN,A,B)。
- Q = quadl(FUN,A,B,TOL)。
- Q = quadl(FUN,A,B,TOL,TRACE)。
- [Q,FCNT] = quadl(...)。

式中各输入参数和输出参数的含义同 quad 函数。

通常 quad 函数具有较快的计算速度，但是准确性较差；而 quadl 函数需要更多的计算时间，但是具有较高的准确性。

3．quadv 函数

有的时候，被积函数 $f(x)$ 是一系列的函数，例如下述积分

$$\int_0^1 x^k \mathrm{d}x, k = 1,2,\cdots,n$$

当 k 取不同的数值时，该积分的结果也不尽相同。针对这种情况，MATLAB 提供了 quadv 函数，可以一次计算多个一元函数的数值积分值。

quadv 函数是 quad 函数的矢量扩展，因此也称为矢量积分。其用法与 quad 相同，分别如下：

- Q = quadv(FUN,A,B)。
- Q = quadv(FUN,A,B,TOL)。
- Q = quadv(FUN,A,B,TOL,TRACE)。
- [Q,FCNT] = quadv(...)。

式中各输入参数和输出参数的含义同 quad 函数。矢量积分的结果是一个向量。

quad、quadl、quadv 3 个函数都要求被积函数 FUN 必须是函数句柄，同时积分限 $[a,b]$ 必须是有限的，因此不能为 inf。此外，在使用上述 3 个函数进行数值积分的求解时，可能会出现如下几种错误信息：

- 'Minimum step size reached'：意味着子区间的长度与计算机舍入误差相当，无法继续计算。原因可能是有不可积的奇点。
- 'Maximum function count exceeded'：意味着积分递归计算超过了 10 000 次。原因可能是有不可积的奇点。

- 'Infinite or Not-a-Number function value encountered'：意味着在积分计算时，区间内出现了浮点数溢出或者被零除的情况。

例 6-23，计算积分 $\int_0^2 \frac{1}{x^3-2x-5}\mathrm{d}x$。

在命令行窗口中输入如下语句：

```
clear all;
F = @(x)1./(x.^3-2*x-5);
Q = quad(F,0,2)
```

输出结果：

```
Q =
  -0.4605
```

6.4.3 多重数值积分

本节将讨论被积函数为二元函数 $f(x,y)$ 和三元函数 $f(x,y,z)$ 的情况。MATLAB 提供了 dblquad 函数和 triplequad 函数分别用于计算二重数值积分和三重数值积分。

1．二重数值积分计算函数 dblquad

dblquad 函数可以用来计算被积函数在矩形区域 $x\in[x_{\min},x_{\max}]$，$y\in[y_{\min},y_{\max}]$ 内的数值积分值。该函数先计算内积分值，然后利用内积分的中间结果来计算二重积分。根据 dxdy 的顺序，称 x 为内积分变量，y 为外积分变量。dblquad 函数的调用格式如下：

- Q = dblquad(FUN,XMIN,XMAX,YMIN,YMAX)。
- Q = dblquad(FUN,XMIN,XMAX,YMIN,YMAX,TOL)。
- Q = dblquad(FUN,XMIN,XMAX,YMIN,YMAX,TOL,@QUADL)。
- Q = dblquad(FUN,XMIN,XMAX,YMIN,YMAX,TOL,MYQUADF)。

式中，FUN 为被积函数的句柄；XMIN、XMAX、YMIN、YMAX 分别为矩形区域在 x 和 y 两个方向上的积分限；TOL 指定绝对计算精度；@QUADL 和 MYQUADF 用以指定计算一维积分时采用的函数，MATLAB 默认采用 quad 函数来计算一维积分；@QUADL 表示用户指定采用 quadl 函数来计算一维积分；MYQUADF 则表示采用用户自己编写的一维积分函数。

例 6-24，计算积分 $\int_0^{\pi}\int_0^{2\pi}(y\sin x+3\cos y-1)\mathrm{d}x\mathrm{d}y$。

在命令行窗口中输入如下语句：

```
clear all;
f=@(x,y)y*sin(x)+3*cos(y)-1;
xmin=pi;
xmax=2*pi;
ymin=0;
ymax=pi;
q=dblquad(f,xmin,xmax,ymin,ymax)
```

输出结果：

```
q =
  -19.7392
```

2．三重数值积分计算函数 triplequad

triplequad 函数可以用来计算被积函数在空间区域 $x \in [x_{min}, x_{max}]$，$y \in [y_{min}, y_{max}]$，$z \in [z_{min}, z_{max}]$内的数值积分值。该函数的调用格式如下：

- Q = triplequad(FUN,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX)。
- Q = triplequad(FUN,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,TOL)。
- Q = triplequad(FUN,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,TOL,@QUADL)。
- Q = triplequad(FUN,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,TOL,MYQUADF)。

上式中相关参数的含义同 dblquad 函数。

三重积分可以用函数 triplequad()来实现，其用法与二重积分类似。

6.5　本章小结

针对数据分析和处理，MATLAB 提供了大量的函数供用户使用。本章介绍了 MATLAB 强大的数据分析和处理功能，主要包括多项式函数、插值函数、数组和函数极限的求解过程及函数的数值积分计算。可以看出利用 MATLAB 进行数据分析非常灵活，读者需要熟练掌握。

第 7 章

符号数学计算



知识要点

符号数学工具箱中的工具建立在功能强大的 Maple 软件的基础上，符号计算式对字符串进行符号分析和运算。符号运算的过程是由 MATLAB 提交请求至 Maple 组件，Maple 组件完成计算以后返回计算值到 MATLAB。符号计算的特点如下：

- 计算以推理解析的方式进行，因此不受计算误差累积所带来的困扰。
- 符号计算可以给出完全正确的封闭解或任意精度的数值解。
- 符号计算指令的调用比较简单，与经典教科书公式相近。
- 计算所需要的时间较长。

学习要求	知识点 \ 学习目标	了解	理解	应用	实践
	符号对象和符号表达式的基本概念		√		
	符号表达式的操作和替换			√	√
	符号函数的操作			√	√
	符号微积分			√	√
	符号积分变换	√			√

7.1　MATLAB 符号计算概述

MATLAB 符号计算是通过集成在 MATLAB 中的符号数学工具箱（Symbolic Math Toolbox）来实现的。和别的工具箱有所不同，该工具箱不是基于矩阵的数值分析，而是使用字符串来进行符号分析与运算。

MATLAB 中的符号数学工具箱是建立在 Maple 基础上的，当进行 MATLAB 符号运算时，它就请求 Maple 软件去计算并将结果返回给 MATLAB，如图 7-1 所示。

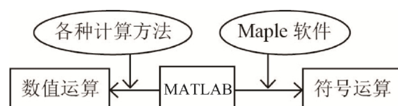


图 7-1　MATLAB 符号运算的工作流程

MATLAB 的符号数学工具箱可以完成几乎所有的符号计算功能。这些功能主要包括：符号表达式的计算，符号表达式的复合、化简，符号矩阵的计算，符号微积分，符号函数画图，符号代数方程求解，符号微分方程求解等。此外，工具箱还支持可变精度运算，即支持符号运算并以指定的精度返回结果。

MATLAB R2018a 集成的符号数学工具箱为 Symbolic Math Toolbox Version 5.8，通过访问 MathWorks 官方网站相应的网址 http://www.mathworks.cn/help/toolbox/symbolic/来获取最新版的符号数学工具箱的最新改进功能。

7.2　符号对象和符号表达式

符号数学工具箱中定义了一种新的数据类型，叫 sym 类。sym 类的实例就是符号对象，符号对象是一种数据结构，是用来存储代表符号的字符串的复杂数据结构。符号表达式是符号变量或常量的组合，在某些特定的情况下，符号变量和符号常量也可以被认为是符号表达式，符号表达式的创建依然使用函数 sym。

7.2.1　符号对象的创建命令

作为符号对象的符号常量、符号变量、符号函数及符号表达式，可以使用函数 sym()、syms()规定和创建，利用 class()函数可以测试建立的操作对象为何种操作对象类型，以及是否为符号对象类型。

1．函数 sym()

函数 sym()的调用格式有如下两种：

- variable＝sym(A,flag)。
- S＝sym('A',flag)。

命令公式由 *A* 来建立一个符号对象 variable，其类型为 sym。如果 *A*（不带单引号）是一个数字、数值矩阵或数值表达式，则输出结果是将数值对象转换成的符号对象。如果 *A*（带单引号）是一个字符串，输出结果则是将字符串转换成的符号对象。

其中 flag 为转换的符号对象应该符合的格式。如果被转换的对象为数值对象，flag 可以有如下选择：

- d：最接近的十进制浮点精确表示。
- e：带（数值计算时）估计误差的有理表示。
- f：十六进制浮点表示。
- r：为默认设置时，最接近有理表示的形式。

对于被转换对象为字符串时，flag 有如下几种选项：

- positive：限定 *A* 为正的实型符号变量。
- real ：限定 *A* 为实型符号变量。

2．函数 syms()

函数 syms()的调用格式如下：

```
syms a b c flag
```

该命令可以建立 3 个或多个符号对象，如 *a*、*b*、*c*。同样，flag 为对转换格式的限定，具体选项同上。

> ○ 提示
>
> 利用 syms 函数创建多个符号变量时，符号变量之间以空格隔开。

7.2.2　符号对象的创建示例

本节将依次介绍符号常量、符号变量、符号表达式、符号函数、符号方程及符号矩阵的创建方法。

例 7-1，符号常量的创建。

在 sym 函数中，如果输入参数为数值常量，则函数返回值为一个符号常量。虽然看上去它是一个数值量，但实际上是一个符号对象。

在命令行窗口中输入：

```
r=sym(2/3)
```

输出结果：

```
r =
  2/3
```

在命令行窗口中输入：

```
f=sym(2/3,'f')
```

输出结果：

```
f =
```

```
   6004799503160661/9007199254740992
```

在命令行窗口中输入：

```
 d=sym(2/3,'d')
```

输出结果：

```
d =
   0.66666666666666662965923251249478
```

在命令行窗口中输入：

```
 e=sym(2/3,'e')
```

输出结果：e =

```
   2/3 - eps/6
```

例 7-2，符号变量的创建。

在符号计算中，符号变量是内容可变的符号对象。通常，符号变量是指一个或多个特定的字符。符号变量的命名规则与 MATLAB 数值计算中变量的命名规则相同，主要包括以下三点：

- 变量名由英文字母开头，可以包括英文字母、数字和下画线。
- 变量名的长度不大于 31 个。
- 字母区分大小写。

MATLAB 使用 sym 函数或 syms 函数来建立符号变量。符号变量名与变量字符串可以是相同的，也可以是不相同的。

在命令行窗口中输入：

```
x=sym('x')
```

输出结果：

```
x=
   x
```

在命令行窗口中输入：

```
y=sym('x')
```

输出结果：

```
y =
   x
```

在命令行窗口中输入：

```
z1=sym('z1','real')
```

输出结果：

```
z1 =
    z1
```

在命令行窗口中输入：

```
z2=sym('z2','positive')
```

输出结果：

```
z2 =
    z2
```

在命令行窗口中输入：

```
assumptions
```

输出结果：

```
ans =
    [ z1 in R_, 0 < z2]
```

在命令行窗口中输入：

```
syms a b c
a,b,c
```

输出结果：

```
    a =
      a
    b =
      b
    c =
      c
```

例 7-3，符号表达式的创建。

在符号计算中，符号表达式是由符号常量、符号变量、符号运算符及专用函数连接起来的符号对象组成的。符号表达式可以分为两类：不带等号的为符号函数；带等号的为符号方程。

MATLAB 使用 sym 函数或 syms 函数来创建符号表达式。

在命令行窗口中输入：

```
f=sym('a*x^4+b*x^3+c*x^2+d*x+e')
```

输出结果：

```
f =
a*x^4 + b*x^3 + c*x^2 + d*x + e
```

在命令行窗口中输入：

```
e=sym('x^2+x^-2=1')
```

输出结果：

```
e=
1/x^2 + x^2 == 1
```

调用 syms 函数来建立表达式。

在命令行窗口中输入：

```
syms a b c d e x
f= a*x^4+b*x^3+c*x^2+d*x+e
```

输出结果：

```
f =
    a*x^4 + b*x^3 + c*x^2 + d*x + e
```

例 7-4，符号矩阵的创建。

元素是符号对象的矩阵叫作符号矩阵。符号矩阵既可以构成符号矩阵函数，也可以构成符号矩阵方程，它们都是符号表达式。通过 syms 函数可以创建符号矩阵。

在命令行窗口中输入：

```
syms x y;
m1=[1,2+x,1;2+x,1,3+y;1,3+y,0]
```

输出结果：

```
m1 =
         [    1, x + 2,    1]
         [ x + 2,    1, y + 3]
         [    1, y + 3,    0]
```

在命令行窗口中输入：

```
m2=sym('[1,2+x,1;2+x,13+y;1,3+y,0]')
```

输出结果：

```
m2 =
         [    1, x + 2,    1]
         [ x + 2,    1, y + 3]
         [    1, y + 3,    0]
```

7.2.3　符号计算中的运算符和函数

自 MATLAB 5.3 集成的 2.1 版本的符号计算工具包开始，MATLAB 采用全新的数据结构、面向对象编程和重载技术，使得符号计算和数值计算在形式和风格上浑然统一。符号计算表达式的运算符和基本函数，无论在形状、名称上，还是在使用方法上，都与数值计算中的运算符和基本函数近乎相同。

以下将介绍符号计算中的运算符号和基本函数。

1．算术运算符号

（1）运算符号“+”“－”“*”“\”“/”“^”分别实现矩阵的加法、减法、乘法、左除、右除和求幂运算。

例 7-5，符号矩阵加法。

在命令行窗口中输入：

```
syms a b c d e f g h;
A=sym('[a,b;c,d]')
A =
     [ a, b]
     [ c, d]
B=sym('[e,f;g,h]')
B =
     [ e, f]
     [ g, h]
R=A+B
```

输出结果：

```
R =
     [ a + e, b + f]
     [ c + g, d + h]
```

例 7-6，符号矩阵左除。

在命令行窗口中输入：

```
syms a b c d e f g h;
A=sym('[a,b;c,d]')
A =
     [ a, b]
     [ c, d]
B=sym('[e,f;g,h]')
B =
     [ e, f]
     [ g, h]
R=A\B
```

输出结果：

```
R =
     [ -(b*g - d*e)/(a*d - b*c), -(b*h - d*f)/(a*d - b*c)]
     [  (a*g - c*e)/(a*d - b*c),  (a*h - c*f)/(a*d - b*c)]
```

（2）运算符号“.*”“.\”“./”“.^”分别实现“元素对元素”的数组乘法、左除、右除和求幂运算。

例 7-7，符号矩阵的乘法与点乘。

在命令行窗口中输入：

```
syms a b c d e f g h;
A=sym('[a,b;c,d]')
A =
     [ a, b]
     [ c, d]
B=sym('[e,f;g,h]')
B =
     [ e, f]
     [ g, h]
R1=A*B
```

输出结果：

```
R1 =
     [ a*e + b*g, a*f + b*h]
     [ c*e + d*g, c*f + d*h]
```

在命令行窗口中输入：

```
R2=A.*B
```

输出结果：

```
R2 =
     [ a*e, b*f]
     [ c*g, d*h]
```

（3）运算符号“'”“.'”分别实现矩阵的共轭转置和非共轭转置。

例 7-8，符号矩阵的转置。

在命令行窗口中输入：

```
syms a b c d;
A=sym('[a,b;c,d]')
A =
     [ a, b]
```

```
      [ c, d]
R1=A'
```

输出结果：

```
R1 =
      [ conj(a), conj(c)]
      [ conj(b), conj(d)]
```

在命令行窗口中输入：

```
R2=A.'
```

输出结果：

```
R2 =
      [ a, c]
      [ b, d]
```

2．关系运算符号

与数值计算中的关系运算符号不同的是，符号计算中的关系运算符号只有以下两种：

（1）运算符号“==”表示对运算符两边的符号对象进行“相等”的比较，返回值为“1”表示相等，返回值为“0”表示不相等。

（2）运算符号“~=”表示对运算符号两边的符号对象进行“不相等”的比较，返回值为“1”表示不相等，返回值为“0”表示相等。

3．指数、对数函数

（1）sqrt、exp、expm 等指数函数在符号计算中的使用方法与数值计算中的使用方法一致。

（2）自 MATLAB 7.x 版本开始，MATLAB 新增加了 log2 函数和 log10 函数，其用法与数值计算中的用法一致。

4．三角函数、双曲函数及其反函数

除 atan2 函数仅能用于数值计算外，其余的三角函数、双曲函数及它们的反函数，无论在数值计算还是在符号计算中，用法一致。

5．复数函数

复数函数包括复数的共轭（conj）、实部（real）、虚部（imag）和模（abs）函数，在数值计算和符号计算中的用法都是一样的。

6．矩阵代数函数

在符号计算中，常用的矩阵代数函数包括：diag 函数、triu 函数、tril 函数、inv 函数、det 函数、rank 函数、rref 函数、null 函数、colspace 函数、poly 函数、expm 函数、eig 函数和 svd 函数。

除了 svd 函数的使用方法有所不同，其余函数的用法与数值计算一致。

例 7-9，符号矩阵的 SVD 分解。

在命令行窗口中输入：

```
f=sym('[1,2,1;2,3,6;1,7,5]')
f =
```

```
        [ 1, 2, 1]
        [ 2, 3, 6]
        [ 1, 7, 5]
    [U,S,V]=svd(f)
```

输出结果：

```
    U =
    [     0.20617816255577867278647750052723,     -0.18281767751192053830238849384246,
0.96128469356079436573565726885 1]
    [    0.60201581377214044365941180414421,       0.79816218560190211775057217364963,
0.022673452393984296851631210788381]
    [     0.77140619990598365798630892894175,     -0.57403381630731667132245685366249,
-0.27462274574817614520424747138197]
    S =
    [ 11.032109063342484567586096292005,   0,     0]
    [                      0, 2.7705627680884633828480975443944,    0]
    [          0,                  0, 0.78520797410533172269772963051585]
    V =
    [     0.19775148863014067403791487260026,     0.30299724195158998284048905286434,
0.93224836825509809153340983805907]
    [         0.690552198317526804477254335957,   -0.71804383401207636383725933546255,
0.086894843552818812855697973076688]
    [    0.69572409052971169492599808156118,       0.62658257540963215504177707204601,
-0.35122964859139975471917772429003]
```

在命令行窗口中输入：

```
    syms x y u v;
    A=sym('[x,y;u,v]')
    A =
        [ x, y]
        [ u, v]
    [U,S,V]=svd(A)
```

输出结果：

```
    Error using mupadmex
    Error in MuPAD command: Nonnumeric entries not handled. [numeric::singularvectors]
    Error in sym/mupadmexnout (line 1734)
        out = mupadmex(fcn,args{:});
    Error in sym/svd (line 49)
     [Usym,Ssym,Vsym] = mupadmexnout('symobj::svdvecs',A);
```

7.2.4　符号对象的类别识别函数

数值计算对象、符号计算对象、字符串作为 MATLAB 中最常使用的数据对象，它们遵循着各自不同的运算法则，但有时在外形上却十分相似。

为管理和使用方便，MATLAB 提供了一些识别不同数据对象的函数，常用的有 class 函数、isa 函数和 whos 函数。

class 函数的返回值 S 是对象 OBJ 的数据类型。S 的值参见表 7-1，其调用格式如下：

```
    S=class(OBJ)
```

表 7-1　数据类型

数据类型	描　述	数据类型	描　述
double	双精度浮点数组	uint8	8 位无符号整型数组
single	单精度浮点数组	int16	16 位有符号整型数组
logical	逻辑数组	uint16	16 位无符号整型数组
char	字符数组	int32	32 位有符号整型数组
cell	单元数组	uint32	32 位无符号整型数组
struct	结构体数组	uint64	64 位无符号整型数组
function_handle	函数句柄	<class_name>	MATLAB 对象的 MATLAB 类名
int8	8 位有符号整型数组	<java_class>	Java 对象的 Java 类名

例 7-10，class 类别识别函数。

在命令行窗口中输入：

```
a=pi;
b='pi';
c=sym('3.1415926');
d=sym('d');
syms e;
classa=class(a)
classb=class(b)
classc=class(c)
classd=class(d)
classe=class(e)
```

输出结果：

```
classa =
        double
classb =
        char
classc =
        sym
classd =
        sym
classe =
        sym
```

isa 函数的调用格式如下：

- isa(OBJ,'classname')：该函数判断输入参数是否是某种类型的对象。函数返回值为 1（true）表示 OBJ 是 classname 类型的；函数返回值为 0（false）表示 OBJ 不是 classname 类型的。

例 7-11，isa 类别识别函数。

在命令行窗口中输入：

```
syms x;
R=isa(x,'sym')
```

输出结果：

```
R =
    1
```

在命令行窗口中输入：

```
R=isa(x,'double')
```

输出结果：

```
R =
    0
```

在命令行窗口中输入：

```
double y;
R=isa(y,'char')
```

输出结果：

```
R =
    0
```

在命令行窗口中输入：

```
R=isa(y,'double')
```

输出结果：

```
R =
    1
```

whos 函数用来列出当前工作区中的变量名及它们的详细信息，如 size（大小）、bytes（字节）及 class（数据类型）等。whos 函数的调用格式如下：

- whos：列出当前工作区内的所有变量及其信息。
- whos VAR1 VAR2：列出当前工作区内的变量 VAR1 和 VAR2 的信息。

例 7-12，whos 类别识别函数。

在命令行窗口中输入：

```
clear
x=0.5;
syms y;
whos
```

输出结果：

```
Name      Size           Bytes  Class     Attributes
x        1x1               8  double
y        1x1              60  sym
```

在命令行窗口中输入：

```
whos y
```

输出结果：

```
Name      Size           Bytes  Class    Attributes
y        1x1              60  sym
```

7.2.5　符号表达式中的变量确定

MATLAB 中的符号对象可以表示符号常量和符号变量。findsym 函数可以帮助用户查找一个符号表达式中的符号变量。该函数的调用格式如下：

- findsym(S)。
- findsym(S,N)。

函数返回符号表达式 S 中的所有符号变量；当指定 N 后，函数返回符号表达式 S 中距离符号变量 x 或 X 最接近的 N 个符号变量。

例 7-13，findsym 变量确定函数。

在命令行窗口中输入：

```
syms x y z u v w
f=sym('3*x^2+2*y+z+u^-1+v^-2+w^-3')
findsym(f)
```

输出结果：

```
ans =
    u,v,w,x,y,z
```

在命令行窗口中输入：

```
findsym(f,1)
```

输出结果：

```
ans =
      x
```

在命令行窗口中输入：

```
findsym(f,2)
```

输出结果：

```
ans =
    x,y
```

在命令行窗口中输入：

```
findsym(f,3)
```

输出结果：

```
ans =
    x,y,w
```

7.2.6 符号精度计算

符号计算的一个非常显著的特点是，由于计算过程中不会出现舍入误差，从而可以得到任意精度的数值解。如果希望计算结果精确，那么就可以牺牲计算时间和存储空间，用符号计算来获得足够高的计算精度。

在符号运算工具箱中有 3 种不同类型的算术运算。

- 数值类型：MATLAB 的浮点算术运算。
- 有理数类型：Maple 的精确符号计算。
- VPA 类型：Maple 的任意精度算术运算。

这 3 种运算各有利弊，在计算时应根据计算精度、时间、存储空间的要求进行合理的选择。浮点运算是最快的运算，需要的计算机内存最小，但是结果不精确。MATLAB 双精度

数输出的数字位数由 format 命令控制，但它内部采用的是由计算机硬件提供的 8 位浮点的表示方法。

符号运算中的有理数算术运算所需要的时间和内存开销都是最大的。只要有足够的内存和足够长的计算时间，总能产生精确的结果。一般符号计算的结果都是字符串，特别是一些符号计算结果从形式上来看是数值，但从变量类型上看，它们仍然是字符串。要从精确解中获得任意精度的解，并改变默认精度，把任意精度符号解变成“真正的”数值解，就需要用到如下几个函数：

（1）函数 digits。

函数 digits()的调用格式如下：

- digits(d)：调用该函数后的近似解的精度变成 *d* 位有效数字。*d* 的默认值是 32 位。

（2）函数 vpa。

函数 vpa()的调用格式如下：

- vpa(A,d)：求符号解 *A* 的近似解，该近似解的有效位数由参数 d 来指定。

（3）函数 double。

函数 double()的调用格式如下：

- double(A)：把符号矩阵或任意精度表示的矩阵 *A* 转换为双精度矩阵。

例 7-14，符号精度示例。

在命令行窗口中输入：

```
sym x;
f=sym('2*x^2+3*x-4')
f =
    2*x^2 + 3*x - 4
s=solve(f)
s =
    41^(1/2)/4 - 3/4
     - 41^(1/2)/4 - 3/4
digits(4);
vpa(s)
```

输出结果：

```
ans =
      0.8508
      -2.351
```

在命令行窗口中输入：

```
vpa(s,6)
```

输出结果：

```
ans =
    0.850781
    -2.35078
```

7.3 符号表达式操作

符号表达式的基本操作包括显示、展开、分解、化简等。这些操作是对数学、物理及各种工程问题进行理论分析时不可缺少的环节。

本节将介绍符号数学工具箱中符号表达式的显示、合并、展开、嵌套、分解和化简操作。

7.3.1 符号表达式显示

在符号表达式的显示过程中，默认采用 MATLAB 式的显示。除了默认的显示方式，MATLAB 符号工具箱中提供了显示函数 pretty，允许用户将符号表达式显示为符合一般数学表达习惯的数学表达式。

例 7-15，符号表达式显示示例。

在命令行窗口中输入：

```
sym x;
f=sym('a*x^3+b*x^2+c*x+d+e*x^-1+f*x^-2+g*x^-3')
pretty(f)
```

输出结果：

```
f=d+c*x+a*x^3+b*x^2+e/x+f/x^2+g/x^3
```

$$d+cx+ax^3+bx^2+\frac{e}{x}+\frac{f}{x^2}+\frac{g}{x^3}$$

7.3.2 符号表达式合并

MATLAB 提供函数实现将符号表达式中的同类项合并的功能，具体调用格式有两种：

- R=collect(S)：将表达式 *S* 中的相同次幂的项合并。其中 *S* 可以是一个表达式，也可以是一个符号矩阵。
- R=collect(S,v)：将表达式 *S* 中 *v* 的相同次幂的项进行合并。如果 *v* 没有指定，则默认将含有 *x* 的相同次幂的项进行合并。

例 7-16，符号表达式合并示例。

在命令行窗口中输入：

```
sym x;
f1=sym('(x-1)^3')
f 1=
     (x - 1)^3
collect(f1)
```

输出结果：

```
ans =
    x^3 - 3*x^2 + 3*x – 1
```

在命令行窗口中输入：

```
syms x y t;
f2=sym('x*cos(t)+y*sin(t)+(x^2+2*x*y+3*y^2)*t')
f2 =
    t*(x^2 + 2*x*y + 3*y^2) + x*cos(t) + y*sin(t)
collect(f2,x)
```

输出结果：

```
ans =
    t*x^2 + (cos(t) + 2*t*y)*x + 3*t*y^2 + sin(t)*y
```

在命令行窗口中输入：

```
collect(f2,y)
```

输出结果：

```
ans =
     (3*t)*y^2 + (sin(t) + 2*t*x)*y + t*x^2 + cos(t)*x
```

7.3.3　符号表达式展开

MATLAB 提供函数实现将表达式展开的功能。它的调用格式如下：

- R = expand(S)：将表达式 *S* 中的各项进行展开。如果 *S* 包含函数，则利用恒等变形将它写成相应的和的形式。该函数多用于多项式，有时也用于三角函数、指数函数和对数函数。

例 7-17，符号表达式展开示例。

在命令行窗口中输入：

```
syms x y;
f1=sym('(x-1)^2*(y-1)')
f1 =
     (x - 1)^2*(y - 1)
expand(f1)
```

输出结果：

```
ans =
    2*x + y - 2*x*y + x^2*y - x^2 – 1
```

在命令行窗口中输入：

```
f2=sym('tan(2*x^2+y)')
f2 =
    tan(2*x^2 + y)
expand(f2)
```

输出结果：

```
ans =
(tan(y) - (2*tan(x^2))/(tan(x^2)^2 - 1))/((2*tan(x^2)*tan(y))/(tan(x^2)^2 - 1) + 1)
```

在命令行窗口中输入：

```
f3=sym('exp((x+y)^2)')
f3 =
    exp((x + y)^2)
expand(f3)
```

输出结果：

```
ans =
    exp(x^2)*exp(y^2)*exp(2*x*y)
```

在命令行窗口中输入：

```
f4=sym('log((x/y)^2)')
f4 =
    log(x^2/y^2)
expand(f4,'IgnoreAnalyticConstraints',true)
```

输出结果：

```
ans =
    2*log(x) - 2*log(y)
```

7.3.4 符号表达式嵌套

MATLAB 提供函数实现将符号表达式转换成嵌套形式的功能。它的调用格式如下：

- R = horner(S)：其中 *S* 是符号多项式矩阵，函数 horner 将其中的每个多项式转换成它们的嵌套形式。

例 7-18，符号表达式嵌套示例。

在命令行窗口中输入：

```
syms x y;
f1=sym('x^3-6*x^2+11*x-6')
f1 =
    x^3 - 6*x^2 + 11*x – 6
horner(f1)
```

输出结果：

```
ans =
    x*(x*(x - 6) + 11) – 6
```

在命令行窗口中输入：

```
f2=sym('[x^2+x;y^3-2*y]')
f2 =
    x^2 + x
    y^3 - 2*y
horner(f2)
```

输出结果：

```
ans =
    x*(x + 1)
    y*(y^2 - 2)
```

7.3.5 符号表达式分解

MATLAB 提供函数实现将符号多项式进行因式分解的功能。它的调用格式如下：

- factor(X)：如果 *X* 是一个多项式或多项式矩阵，系数是有理数，那么该函数将把 *X* 表示成系数为有理数的低阶多项式相乘的形式；如果 *X* 不能分解成有理多项式乘

积的形式，则返回 *X* 本身。

例 7-19，符号表达式分解示例。

在命令行窗口中输入：

```
syms x y;
f1=sym('2*x^2-7*x*y-22*y^2-5*x+35*y-3')
f1 =
    2*x^2 - 7*x*y - 5*x - 22*y^2 + 35*y – 3
factor(f1)
```

输出结果：

```
ans =
     (2*x - 11*y + 1)*(x + 2*y - 3)
```

在命令行窗口中输入：

```
f2=sym('[2*x^2-5*x*y-3*y^2;x^3-y^3]')
f2 =
    2*x^2 - 5*x*y - 3*y^2
    x^3 - y^3
factor(f2)
```

输出结果：

```
ans =
     (x - 3*y)*(2*x + y)
     (x - y)*(x^2 + x*y + y^2)
```

在命令行窗口中输入：

```
f3=sym('12345678901234567890')
f3 =
    12345678901234567890
factor(f3)
```

输出结果：

```
ans =
    2*3^2*5*101*3541*3607*3803*27961
```

7.3.6　符号表达式简化

MATLAB 根据一定的规则对符号表达式进行简化，简化的函数包含 simplify 和 simple。

（1）simplify 函数的调用格式如下：

- R= simplify(S)：该函数是一个强有力的具有普遍意义的工具。它应用于包含和式、方根、分数的乘方、指数函数、对数函数、三角函数、Bessel 函数及超越函数等的表达式，并利用 Maple 化简规则对表达式进行简化。其中 ***S*** 可以是符号表达式矩阵。

（2）simple 函数调用格式如下：

- R = simple(S)：用几种不同的算术简化规则对符号表达式进行简化，返回时表达式 *S* 变得简短。如果 ***S*** 是符号表达式矩阵，则返回整个矩阵变成最短的形式，而不一定使每一项都最短；如果不给定输出参数 ***R***，该函数将显示所有使表达式 *S* 变短的

简化形式，并返回其中最短的一个。

- [R, how] = simple(S)：不显示简化的中间结果，只显示寻找到的最短形式及找到该形式所用的简化方法。返回值中，R是符号表达式，how是一个描述简化方法的字符串。

例 7-20，符号表达式化简示例。

在命令行窗口中输入：

```
sym x;
f1=sym('(x^3-1)/(x-1)')
f1 =
    (x^3 - 1)/(x - 1)
simplify(f1)
```

输出结果：

```
ans =
    x^2 + x + 1
```

在命令行窗口中输入：

```
f2=sym('cos(x)+i*sin(x)')
f2 =
    cos(x) + sin(x)*i
simple(f2)
```

输出结果：

```
simplify:
    cos(x) + sin(x)*i
radsimp:
    cos(x) + sin(x)*i
simplify(100):
    cos(x) + sin(x)*i
combine(sincos):
    cos(x) + sin(x)*i
combine(sinhcosh):
    cos(x) + sin(x)*i
combine(ln):
    cos(x) + sin(x)*i
factor:
    cos(x) + sin(x)*i
expand:
    cos(x) + sin(x)*i
combine:
    cos(x) + sin(x)*i
rewrite(exp):
    exp(x*i)
rewrite(sincos):
    cos(x) + sin(x)*i
rewrite(sinhcosh):
    cosh(x*i) + sinh(x*i)
rewrite(tan):
    (tan(x/2)*2*i)/(tan(x/2)^2 + 1) - (tan(x/2)^2 - 1)/(tan(x/2)^2 + 1)
mwcos2sin:
    sin(x)*i - 2*sin(x/2)^2 + 1
collect(x):
    cos(x) + sin(x)*i
```

```
ans =
    exp(x*i)
```

矩阵的基本操作主要有矩阵的构建、矩阵维度与矩阵大小的改变、矩阵的索引、矩阵属性信息的获取、矩阵结构的改变等。对于这些操作，MATLAB 中都有固定指令或相应的库函数与之相对应。

7.4　符号表达式替换

在处理一些结构较为复杂、变量较多的数学模型时，引入一些新的变量进行代换，以简化其结构，从而达到解决问题的目的，这种方法叫作变量代换法。

例如，求不定积分 $\int \frac{1}{t\left(t^7+2\right)}\mathrm{d}t$，设 $x=\frac{1}{t}$，则

$$\int \frac{1}{t\left(t^7+2\right)}\mathrm{d}t = -\int \frac{x^6}{1+2x^7}\mathrm{d}x = -\frac{1}{14}\ln\left|1+2x^7\right| + c = -\frac{1}{14}\ln\left|2+t^7\right| + \frac{1}{2}\ln|t| + c$$

变量代换法是一种非常有效的解题方法，尤其是处理一些复杂的不等式问题时，效果明显。合理代换往往能简化题目的信息，凸显隐含条件，沟通量与量之间的关系，对发现解题思路、优化解题过程有着重要的作用。

MATLAB 提供了 subs 函数和 subexpr 函数进行变量代换，或者叫作符号表达式的替换。subs 函数利用符号变量或符号表达式替换目标符号表达式中的符号变量（包括符号常量）；subexpr 函数利用符号变量替换目标符号表达式中的某个子符号表达式。

7.4.1　subs 替换函数

函数 subs 可以用指定符号替换符号表达式中的某一特定符号。它的调用格式如下：

- R = subs(S)：用工作区中的变量值替代符号表达式 S 中的所有符号变量。如果没有指定某符号变量的值，则返回值中该符号变量不被替换。
- R = subs(S,New)：用新符号变量 New 替代原来符号表达式 S 中的默认变量。确定默认变量的规则与函数 findsym 的规则相同。
- R = subs(S,Old,New)：用新符号变量 New 替代原来符号表达式 S 中的变量 Old。当 New 是数值形式的符号时，实际上用数值代替原来的符号来计算表达式的值，只是所得结果仍然是字符串形式。

例 7-21，subs 替换函数示例。

在命令行窗口中输入：

```
syms x y;
f=sym('x^2+x*y+y^2')
f =
    x^2 + x*y + y^2
x=2
subs(f)
```

输出结果：

```
ans =
    y^2 + 2*y + 4
```

在命令行窗口中输入：

```
y=2
subs(f)
```

输出结果：

```
ans =
    12
```

在命令行窗口中输入：

```
subs(f,'t^2')
```

输出结果：

```
ans =
    t^4 + t^2*y + y^2
```

在命令行窗口中输入：

```
subs(f,{'x','y'},{3,4})
```

输出结果：

```
ans =
    37
```

7.4.2 subexpr 替换函数

函数 subexpr 将表达式中重复出现的字符串用变量代替，它的调用格式如下：

- [Y,SIGMA] = subexpr(S,SIGMA)：指定用变量 SIGMA 的值（必须为符号对象）来代替符号表达式（可以是矩阵）中重复出现的字符串。替换后的结果由 *Y* 返回，被替换的字符串由 SIGMA 返回。
- [Y,SIGMA] = subexpr(S,'SIGMA')：这种形式和上一种形式的不同在于第二个输入参数是字符或字符串，它用来替换符号表达式中重复出现的字符串。其他参数与上面的形式相同。

例 7-22，subexpr 替换函数示例。

在命令行窗口中输入：

```
syms x a;
f=sym('2*x^2+a*x+3+x^-1')
s=solve(f)
```

输出结果：

```
    s = (a^2/36 - 1/2)/(a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 - 1/2)^3)^(1/2) - a^3/216
- 1/4)^(1/3) - a/6 + (a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 - 1/2)^3)^(1/2) - a^3/216
- 1/4)^(1/3) - a/6 - (a^2/36 - 1/2)/(2*(a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 - 1/2)^3)^(1/2)
- a^3/216 - 1/4)^(1/3)) - (a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 - 1/2)^3)^(1/2) - a^3/216
- 1/4)^(1/3)/2 + (3^(1/2)*((a^2/36 - 1/2)/(a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 -
1/2)^3)^(1/2) - a^3/216 - 1/4)^(1/3) - (a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 -
```

```
1/2)^3)^(1/2) - a^3/216 - 1/4)^(1/3))*i)/2- a/6 - (a^2/36 - 1/2)/(2*(a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 - 1/2)^3)^(1/2) - a^3/216 - 1/4)^(1/3)) - (a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 - 1/2)^3)^(1/2) - a^3/216 - 1/4)^(1/3)/2 - (3^(1/2)*((a^2/36 - 1/2)/(a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 - 1/2)^3)^(1/2) - a^3/216 - 1/4)^(1/3) - (a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 - 1/2)^3)^(1/2) - a^3/216 - 1/4)^(1/3))*i)/2
```

继续输入：

```
r=subexpr(s)
```

输出结果：

```
sigma =
    a/8 + ((a^3/216 - a/8 + 1/4)^2 - (a^2/36 - 1/2)^3)^(1/2) - a^3/216 - 1/4
r =
    (a^2/36 - 1/2)/sigma^(1/3) - a/6 + sigma^(1/3) - a/6 + (3^(1/2)*((a^2/36 - 1/2)/sigma^(1/3) - sigma^(1/3))*i)/2 - (a^2/36 - 1/2)/(2*sigma^(1/3)) - sigma^(1/3)/2 - a/6 - (3^(1/2)*((a^2/36 - 1/2)/sigma^(1/3) - sigma^(1/3))*i)/2 - (a^2/36 - 1/2)/(2*sigma^(1/3)) - sigma^(1/3)/2
```

7.5　符号函数的操作

MATLAB 具有对符号表达式执行更高级运算的功能。MATLAB 提供了把两个符号函数复合成一个符号函数的功能函数，同时也提供了对符号函数求函数表达式的逆功能函数等。

本节讲解符号函数的操作，主要包含符号函数的复合运算及符号函数的求逆运算。复合运算的功能函数为 compose，求逆运算的功能函数为 finverse。

7.5.1　复合函数操作

在 MATLAB 中，符号表达式的复合函数运算主要是通过 compose 函数来实现的，该函数的调用格式如下：

- compose(f,g)：返回复合函数 $f(g(y))$，此处，$f=f(x)$，$g=g(y)$。其中，x 是 findsym 定义的 f 函数的符号变量，y 是 findsym 定义的 g 函数的符号变量。
- compose(f,g,x,z)：返回自变量为 z 的复合函数 $f(g(z))$，并且使 x 成为 f 函数的独立变量。

例 7-23，compose 复合运算函数示例。

在命令行窗口中输入：

```
syms x y;
f=sym('x+x^-1')
f =
    x + 1/x
g=sym('sin(x)')
g =
    sin(x)
h=sym('1+y^2')
h =
    y^2 + 1
compose(f,g)
```

输出结果：

```
ans =
    sin(x) + 1/sin(x)
```

在命令行窗口中输入：

```
compose(g,f)
```

输出结果：

```
ans =
    sin(x + 1/x)
```

在命令行窗口中输入：

```
compose(f,h,'x','t')
```

输出结果：

```
ans =
    1/(t^2 + 1) + t^2 + 1
```

7.5.2 反函数操作

在 MATLAB 中，符号表达式的反函数运算主要是通过函数 finverse 来实现的。finverse 函数的调用格式如下：

- g＝flnverse(f)：返回符号函数 *f* 的反函数 *g*。其中，*f* 是一个符号函数表达式，其变量为 *x*。求得的反函数 *g* 是一个满足 $g(f(x))=x$ 的符号函数。
- g＝flnverse(f,v)：返回自变量为 *v* 的符号函数 *f* 的反函数。求得的反函数 *g* 是一个满足 $g(f(v))=v$ 的符号函数。当 *f* 包含不止一个符号变量时，往往使用这种求反函数的调用格式。

例 7-24，finverse 反函数示例。

在命令行窗口中输入：

```
sym x,y;
f1=sym('1/(sin(x)+cos(x))')
f1 =
    1/(cos(x) + sin(x))
finverse(f1)
```

输出结果：

```
ans =
    -log((2^(1/2)*(x^2*(-2*i) + i)^(1/2) + 1 + i)/(2*x))*i
```

在命令行窗口中输入：

```
f2=sym('x^2+2*x*y+y^2')
f2 =
    x^2 + 2*x*y + y^2
finverse(f2,y)
```

输出结果：

```
ans =
    y^(1/2) - x
```

7.6　符号微积分

微积分运算在数学计算中的重要性是不言而喻的，整个高等数学就是建立在微积分运算的基础上的。同时微积分运算也是后面求解符号微分方程的必要知识储备。

在符号数学工具箱中提供了一些常用的函数来支持具有重要基础意义的微积分运算，涉及微分、求极限、积分、级数求和和泰勒级数等。下面将具体介绍符号运算在微积分中的使用方法。

7.6.1　符号表达式的极限

求微分的基本思想是当自变量趋近某个值时，求函数值的变化。“无穷逼近”是微积分的一个基本思想，求极限是非常普遍的。事实上，导数就是由极限给出的：

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

在 MATLAB 中，用函数 limit 来求表达式的极限。函数 limit 的调用格式如下：

- limit(F,x,a)：求当 $x\to a$ 时，符号表达式 F 的极限。
- limit(F,a)：符号表达式 F 采用默认自变量（可由函数 findsym 求得），该函数求 F 的自变量趋近于 a 时的极限值。
- limit(F)：符号表达式 F 采用默认自变量，并以 $a=0$ 作为自变量的趋近值，从而求符号表达式 F 的极限值。
- limit(F,x,a,'right')或 limit(F,x,a,'left')：分别求取符号表达式 F 的左极限和右极限，即自变量从左边或右边趋近于 a 时的函数极限值。

例 7-25，符号表达式极限示例。

在命令行窗口中输入：

```
syms x;
f1=sym((cos(x)+sin(x)-x)/x)
f1 =
     (cos(x) - x + sin(x))/x
limit(f,x,inf)
```

输出结果：

```
ans =
    -1
```

在命令行窗口中输入：

```
limit(f,x,-inf)
```

输出结果：

```
ans =
    -1
```

在命令行窗口中输入：

```
limit(f,x,0)
```

输出结果：

```
ans =
    NaN
```

在命令行窗口中输入：

```
f2=sym('(sin(x)-x)/x')
f2 =
    -(x - sin(x))/x
limit(f2,x,0,'right')
```

输出结果：

```
ans =
    0
```

在命令行窗口中输入：

```
limit(f2,x,0,'left')
```

输出结果：

```
ans =
    0
```

7.6.2 符号表达式的微分

MATLAB 提供的函数可以完成一元及多元符号表达式函数的各阶微分，功能函数 diff 可以完成一元或多元函数的任意阶数的微分。对于自变量的个数多于一个的符号矩阵，微分为 Jocabian 矩阵，采用功能函数 jacobian 实现微分。

1. diff 函数

当创建符号表达式后，就可以利用函数 diff 对它进行微分运算。函数 diff 的调用格式如下：

- diff(S,'v')：将符号“*v*”视作变量，对符号表达式或符号矩阵 ***S*** 求取微分。
- diff(S,n)：将 ***S*** 中的默认变量进行 ***n*** 阶微分运算，其中默认变量可以用 findsym 函数确定，参数 ***n*** 必须是正整数。
- diff(S,'v',n)：将符号“*v*”视作变量，对符号表达式或矩阵 ***S*** 进行 ***n*** 阶微分运算。

2. jacobian 函数

函数 jacobian 的调用格式如下：

- R = jacobian(w,v)：其中 ***w*** 是一个符号列向量，***v*** 是指定进行变换的变量所组成的行向量。

例 7-26，符号表达式微分示例。

在命令行窗口中输入：

```
syms x y z;
f1=sym('exp(x*sin(y))+log(z)')
f1 =
    exp(x*sin(y)) + log(z)
```

```
diff(f1,x)
```

输出结果：

```
ans =
    exp(x*sin(y))*sin(y)
```

在命令行窗口中输入：

```
diff(f1,y)
```

输出结果：

```
ans =
    x*exp(x*sin(y))*cos(y)
```

在命令行窗口中输入：

```
diff(f1,z)
```

输出结果：

```
ans =
    1/z
```

在命令行窗口中输入：

```
diff(f1,x,2)
```

输出结果：

```
ans =
    exp(x*sin(y))*sin(y)^2
```

在命令行窗口中输入：

```
f2=sym('[x^2+y^2;y*z]')
f2 =
    x^2 + y^2
        y*z
J=jacobian(f2,[x,y])
```

输出结果：

```
J =
    [ 2*x, 2*y]
[   0,   z]
```

7.6.3　符号表达式的积分

在数学中，积分和微分是一对互逆的运算。符号数学工具箱中提供了函数 int 来求符号表达式的积分，它的调用格式如下：

- R=int(S)：用默认变量求符号表达式 *S* 的不定积分，默认变量可用函数 findsym 确定。
- R = int(S,v)：用符号标量 *v* 作为变量求符号表达式 *S* 的不定积分值。
- R = int(S,a,b)：符号表达式采用默认变量，该函数求默认变量从 *a* 变到 *b* 时符号表达式 *S* 的定积分值。如果 *S* 是符号矩阵，那么积分将对各个元素分别进行，而且每个元素的变量也可以独立地由函数 findsym 来确定，*a* 和 *b* 可以是符号或数值标量。

- R = int(S,v,a,b)：符号表达式采用符号标量 *v* 作为标量，求当 *v* 从 *a* 变到 *b* 时，符号表达式 *S* 的定积分值。其他参数和上一种调用方式相同。

例 7-27，符号表达式积分示例。

在命令行窗口中输入：

```
syms x,y;
f1=sym('x+x^-1')
f1 =
    x + 1/x
int(f1)
```

输出结果：

```
ans =
    log(x) + x^2/2
```

在命令行窗口中输入：

```
f2=sym('x*y+(x*y)^-1')
f2 =
    x*y+(x*y)^-1
int(f2,y)
```

输出结果：

```
ans =
    log(y)/x + (x*y^2)/2
```

在命令行窗口中输入：

```
int(f1,1,2)
```

输出结果：

```
ans =
    log(2) + 3/2
```

在命令行窗口中输入：

```
int(f2,y,1,2)
```

输出结果：

```
ans =
     (3*x)/2 + log(2)/x
```

7.6.4 符号表达式的级数求和

MATLAB 提供的函数 symsum 用于对符号表达式进行求和。该函数的调用格式如下：

- r = symsum(s,a,b)：求符号表达式 *s* 中默认变量从 *a* 变到 *b* 时的有限和。
- r = symsum(s,v,a,b)：求符号表达式 *s* 中变量 *v* 从 *a* 变到 *b* 时的有限和。

例 7-28，符号表达式级数求和示例。

在命令行窗口中输入：

```
syms x y n;
f1=sym('x^2')
f1 =
    x^2
```

```
symsum(f1,0,n-1)
```

输出结果：

```
ans =
    (n*(2*n - 1)*(n - 1))/6
```

在命令行窗口中输入：

```
f2=sym('x^n')
f2 =
    x^n
symsum(f2,n,0,inf)
```

输出结果：

```
ans =
    piecewise([1 <= x, Inf], [abs(x) < 1, -1/(x - 1)])
```

7.6.5 符号表达式的泰勒级数

MATLAB 提供函数 taylor 用来求符号表达式的泰勒级数展开式，该函数的调用格式如下：

- r = taylor(f)：*f* 是符号表达式，其变量采用默认变量，该函数将返回 *f* 在变量等于 0 处作 5 阶泰勒展开时的展开式。
- r = taylor(f,n,v)：符号表达式 *f* 以符号标量 *v* 作为自变量，返回 *f* 的 *n*-1 阶麦克劳林级数（即在 *v* =0 处作泰勒展开）展开式。
- r = taylor(f,n,v,a)：返回符号表达式 *f* 在 *v* = *a* 处作 *n*-1 阶泰勒展开时的展开式。

例 7-29，符号表达式的泰勒级数展开示例。

在命令行窗口中输入：

```
syms x,y;
f1=sym('sin(x)/(2+sin(x))')
f1 =
    sin(x)/(sin(x) + 2)
taylor(f1)
```

输出结果：

```
ans =
    - (13*x^5)/480 + x^4/48 + x^3/24 - x^2/4 + x/2
```

在命令行窗口中输入：

```
f2=sym('sin(x)+1/cos(y)')
f2 =
    sin(x) + 1/cos(y)
taylor(f2,8,y)
```

输出结果：

```
ans =
    (61*y^6)/720 + (5*y^4)/24 + y^2/2 + sin(x) + 1
```

在命令行窗口中输入：

```
taylor(f2,4,x,1)
```

输出结果：

```
ans =
    sin(1) - (sin(1)*(x - 1)^2)/2 + 1/cos(y) + cos(1)*(x - 1) - (cos(1)*(x - 1)^3)/6
```

7.7 符号积分变换

在数学中，为了把较复杂的运算转换为比较简单的运算，经常采用一种变换手段。例如数量的乘积或商可以变换成对数的和或差，然后再取反对数即可求得原来数量的乘积或商。这一变换方法的目的就是把比较复杂的乘除运算通过对数变换转换为简单的加减运算。

所谓积分变换，就是通过积分运算，把一类函数 *A* 变换成另一类函数 *B*，函数 *B* 一般是含有参量 *α* 的积分：$\int_a^b f(t)K(t,\alpha)\mathrm{d}t$。这一变换的目的就是把某函数类 *A* 中的函数 $f(t)$通过积分运算变成另一类函数 *B* 中的函数 $F(\alpha)$。这里 $K(t,\alpha)$是一个确定的二元函数，叫作积分变换的核。当选取不同的积分区间与变换核时，就成为不同的积分变换。$f(t)$叫作原函数，$F(\alpha)$叫作象函数。在一定条件下，原函数与象函数两者一一对应，成为一个积分变换对。变换是可逆的，由原函数求象函数叫作正变换，反之则是逆变换。

积分变换的理论与方法，在自然科学与工程技术的各个领域中都有着极其广泛的应用，成为不可缺少的运算工具。变换的使用，会极大地简化计算，有的变换则为开创新的学科奠定了基础。

7.7.1 傅里叶变换及其反变换

时域中的 $f(t)$与它在频域中的 Fourier 变换 $F(\omega)$之间存在如下关系：

$$F(\omega)=\int_{-\infty}^{\infty} f(t)\mathrm{e}^{-j\omega t}\mathrm{d}t$$

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\mathrm{e}^{j\omega t}\mathrm{d}\omega$$

由计算机完成这种变换的途径有两条：一是直接调用指令 fourier 和 ifourier 进行；二是根据上面的定义，利用积分指令 int 实现。下面只介绍 fourier 和 ifourier 的使用及相关注意事项，至于如何根据定义求变换，请读者自己完成。

- Fw = fourier(ft,t,w)：求时域函数 ft 的 Fourier 变换 Fw，ft 是以 *t* 为自变量的时域函数，Fw 是以圆频率 *w* 为自变量的频域函数。
- ft = ifourier(Fw,w,t)：求频域函数 Fw 的 Fourier 反变换 ft，ft 是以 *t* 为自变量的时域函数，Fw 是以圆频率 *w* 为自变量的频域函数。

例 7-30，傅里叶变换示例。

在命令行窗口中输入：

```
syms t w real;
f=sym('cos(t)*sin(t)')
f =
    cos(t)*sin(t)
```

```
fourier(f,t,w)
```

输出结果：

```
ans =
    -(pi*(dirac(w - 2) - dirac(w + 2))*i)/2
```

在命令行窗口中输入：

```
ifourier(ans,w,t)
```

输出结果：

```
ans =
     (exp(-t*2*i)*i)/4 - (exp(t*2*i)*i)/4
```

在命令行窗口中输入：

```
simplify(ans)
```

输出结果：

```
ans =
    sin(2*t)/2
```

7.7.2　拉普拉斯变换及其反变换

Laplace 变换及其反变换的定义为：

$$F(s)=\int_0^{\infty} f(t)\mathrm{e}^{-st}\mathrm{d}t$$

$$f(t)=\frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty} F(s)\mathrm{e}^{st}\mathrm{d}s$$

与 Fourier 变换相似，Laplace 变换与反变换的实现也有两条途径：直接调用指令 laplace 和 ilaplace；或者根据上面的定义，利用积分指令 int 实现。相较而言，直接使用 laplace 和 ilaplace 指令实现变换较为简洁。具体的调用格式如下：

- Fs = laplace(ft,t,s)：求时域函数 ft 的 Laplace 变换 Fs，ft 是以 t 为自变量的时域函数，Fs 是以复频率 s 为自变量的频域函数。
- ft = ilaplace(Fs,s,t)：求频域函数 Fs 的 Laplace 反变换 ft，ft 是以 t 为自变量的时域函数，Fs 是以复频率 s 为自变量的频域函数。

例 7-31，拉普拉斯变换示例。

在命令行窗口中输入：

```
syms s t;
syms a positive;
f=sym('exp(2*t)+5*dirac(a-t)')
f =
    exp(2*t) + 5*dirac(a - t)
laplace(f,t,s)
```

输出结果：

```
ans =
    5*exp(-a*s) + 1/(s - 2)
```

在命令行窗口中输入：

```
ilaplace(ans,s,t)
```

输出结果：

```
ans =
    exp(2*t) + 5*dirac(a - t)
```

7.7.3 Z 变换及其反变换

一个离散因果序列的 Z 变换及其反变换定义为：

$$F(z)=\sum_{n=0}^{\infty} f(n)z^{-n}$$

$$F(n)=z^{-1}\{F(z)\}$$

涉及 Z 反变换具体计算的常见方法有 3 种，分别是幂级数展开法、部分分式展开法和围线积分法。MATLAB 的符号数学工具箱中采用围线积分法设计求 Z 反变换的 iztrans 指令，相应的数学表达式是：

$$f(n)=\frac{1}{2\pi j}\oint F(z)z^{n-1}\mathrm{d}z$$

具体的调用格式如下：

- FZ = ztrans(fn)：求时域函数 fn 的 Z 变换 FZ，默认 fn 是变量 n 的函数，生成的 Z 变换是以复频率 z 为变量的函数。
- FZ = ztrans(fn,w)：求时域函数 fn 的 Z 变换 FZ，默认 fn 是变量 n 的函数，生成的 Z 变换是以变量 w 来代替复频率 z 的函数。
- FZ = ztrans(fn,n,z)：求时域函数 fn 的 Z 变换 FZ，fn 是以 n 为自变量的时域序列，FZ 是以复频率 z 为自变量的频域函数。
- fn = iztrans(FZ,z,n)：求频域函数 FZ 的 Z 反变换 fn，fn 是以 n 为自变量的时域序列，FZ 是以复频率 z 为自变量的频域函数。

例 7-32，Z 变换示例。

在命令行窗口中输入：

```
syms n;
f=sym('n^3+n^2+n+1')
f =
    n^3 + n^2 + n + 1
ztrans(f)
```

输出结果：

```
ans =
    z/(z - 1) + z/(z - 1)^2 + (z^2 + z)/(z - 1)^3 + (z^3 + 4*z^2 + z)/(z - 1)^4
```

在命令行窗口中输入：

```
clear
syms a b n w;
f=sym('sin(a*n)+cos(b*n)')
```

```
    f =
        cos(b*n) + sin(a*n)
    ztrans(f,w)
```

输出结果：

```
    ans =
    (w*(w - cos(b)))/(w^2 - 2*cos(b)*w + 1) + (w*sin(a))/(w^2 - 2*cos(a)*w + 1)
```

在命令行窗口中输入：

```
    ztrans(f,a,w)
```

输出结果：

```
    ans =
         (w*cos(b*n))/(w - 1) + (w*sin(n))/(w^2 - 2*cos(n)*w + 1)
```

在命令行窗口中输入：

```
    iztrans(ans,w,n)
```

输出结果：

```
    ans =
    sin(n^2) - cos(b*n)*(kroneckerDelta(n, 0) - 1) + cos(b*n)*kroneckerDelta(n, 0)
```

7.8　符号代数方程求解

从学习代数开始，我们就一直在探索关于方程的求解理论，从最初的代入消元法和加减消元法，到数值计算中的牛顿迭代法、高斯消元法，一直到微分方程的求解理论，方程在数学中的重要性就包含在这漫长的探索、深化过程中。

由于代数方程只涉及符号对象的代数运算，相对比较简单，它还可以细分为线性方程和非线性方程两类。前者往往可以很容易地求得所有解，但是对于后者来说，却经常容易丢掉一些解，这时就必须借助函数绘制图形，通过图形来判断方程解的个数。

这里所讲的一般代数方程包括线性、非线性和超越方程等，求解指令是 solve。当方程组不存在符号解，又无其他自由参数时，solve 将给出数值解。该指令的调用方式包括以下几种：

- g = solve(eq)：其中 eq 可以是符号表达式或不带符号的字符串，该函数用于求解方程 eq =0，其自变量采用默认变量，可以通过函数 findsym 来确定。
- g = solve(eq,var)：求解方程 eq = 0，其自变量由参数 var 指定。其中 eq 和上一种调用方式相同。返回值 ***g*** 是由方程的所有解构成的列向量。
- g = solve(eq1,eq2,⋯,eqn)：求解由符号表达式或不带符号的字符串 eq1,eq2,⋯,eqn 组成的方程组。其中的自变量为整个方程组的默认变量，即将函数 findsym 作用于整个方程组时返回的变量。
- g = solve(eq1,eq2,⋯,eqn,var1,var2,⋯,varn)：求解由符号表达式或不带等号的字符串 eq1,eq2,⋯,eqn 组成的方程组。其自变量由输入参数 var1,var2,⋯,varn 指定。

对于上面的 4 种调用方式，输出的解有以下 3 种情况：

- 对于单个方程单个输出参数的情况，将返回由多个解构成的列向量。

- 对于有和方程数目相同的输出参数的情况，方程组的解将分别赋给每个输出参数，并按照字母表的顺序进行排列。
- 对于只有一个输出参数的方程组，方程组的解将以结构矩阵的形式赋给输出参数。

例 7-33，符号代数方程求解示例。

在命令行窗口中输入：

```
syms x;
f=sym('a*x+b*x^-1+c')
f =
    c + a*x + b/x
solve(f)
```

输出结果：

```
ans =
    -(c + (c^2 - 4*a*b)^(1/2))/(2*a)
    -(c - (c^2 - 4*a*b)^(1/2))/(2*a)
```

在命令行窗口中输入：

```
clear
syms a x;
f=sym('a*x^2+b*x+c')
f =
    a*x^2 + b*x + c
solve(f,x)
```

输出结果：

```
ans =
    -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
    -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

在命令行窗口中输入：

```
solve(f,a)
```

输出结果：

```
ans =
    -(c + b*x)/x^2
```

在命令行窗口中输入：

```
clear
syms x y;
f1=sym('x^2+y^2=25')
f1 =
x^2 + y^2 == 25
f2=sym('x*y=12')
f2 =
    x*y == 12
[x,y]=solve(f1,f2)
```

输出结果：

```
x =
    4
    3
```

```
        -3
        -4
    y =
        3
        4
        -4
        -3
```

在命令行窗口中输入：

```
    clear
    syms x y a b;
    f1=sym('x^2+y^2=a^2')
    f1 =
        x^2 + y^2 == a^2
    f2=sym('x*y=b')
    f2 =
        x*y ==b
    S=solve(f1,f2,x,y)
```

输出结果：

```
    S =
    x: [4x1 sym]
    y: [4x1 sym]
```

在命令行窗口中输入：

```
    S.x
```

输出结果：

```
    ans =
    -((a^2/2 + (-(- a^2 + 2*b)*(a^2 + 2*b))^(1/2)/2)^(3/2) - a^2*(a^2/2 + (-(- a^2 + 2*b)*(a^2 +
2*b))^(1/2)/2)^(1/2))/b
    ((a^2/2 + (-(- a^2 + 2*b)*(a^2 + 2*b))^(1/2)/2)^(3/2) - a^2*(a^2/2 + (-(- a^2 + 2*b)*(a^2 +
2*b))^(1/2)/2)^(1/2))/b
    -((a^2/2 - (-(- a^2 + 2*b)*(a^2 + 2*b))^(1/2)/2)^(3/2) - a^2*(a^2/2 - (-(- a^2 + 2*b)*(a^2 +
2*b))^(1/2)/2)^(1/2))/b
    ((a^2/2 - (-(- a^2 + 2*b)*(a^2 + 2*b))^(1/2)/2)^(3/2) - a^2*(a^2/2 - (-(- a^2 + 2*b)*(a^2 +
2*b))^(1/2)/2)^(1/2))/b
```

在命令行窗口中输入：

```
    S.y
```

输出结果：

```
    ans =
    (a^2/2 + (-(- a^2 + 2*b)*(a^2 + 2*b))^(1/2)/2)^(1/2)
    -(a^2/2 + (-(- a^2 + 2*b)*(a^2 + 2*b))^(1/2)/2)^(1/2)
    (a^2/2 - (-(- a^2 + 2*b)*(a^2 + 2*b))^(1/2)/2)^(1/2)
    -(a^2/2 - (-(- a^2 + 2*b)*(a^2 + 2*b))^(1/2)/2)^(1/2)
```

7.9 符号微分方程求解

从数值计算角度看，与初值问题求解相比，微分方程边值问题的求解显得复杂和困难。对于应用数学工具去求解实际问题的科研人员来说，此时不妨通过符号计算指令进行求解尝试。因为对于符号计算来说，不论是初值问题还是边值问题，其求解微分方程的指令形式都

相同，且相当简单。

当然，符号计算可能花费较多的计算机资源，可能得不到简单的解析解或封闭形式的解，甚至无法求解。既然没有万能的微分方程一般解法，那么，求解微分方程的符号法和数值法就有很好的互补作用。

函数 dsolve 用来求常微分方程的符号解。在方程中，用大写字母 D 表示一次微分，D2、D3 分别表示二次、三次微分运算。以此类推，符号 D2y 表示 $\frac{d^2y}{dt^2}$。函数 dsolve 把 *d* 后面的字符当作因变量，并默认所有这些变量对符号 *t* 进行求导。函数 dsolve 的调用格式如下：

- r = dsolve('eq1,eq2,⋯','cond1,cond2,⋯','v')：求由 eq1,eq2,⋯指定的常微分方程的符号解。常微分方程以变量 *v* 作为自变量，参数 cond1,cond2,⋯用于指定方程的边界条件或者初始条件。如果 *v* 不指定，将默认 *t* 为自变量。
- r = dsolve('eq1','eq2',⋯,'cond1','cond2',⋯,'v')：求由 eq1,eq2,⋯指定的常微分方程的符号解。这些常微分方程都以 *v* 作为自变量。这些单独输入的方程的最大允许个数为 12。其他参数与上一种调用方式相同。微分方程的初始条件或边界条件都以变量 *v* 作为自变量，其形式为 $y(a)=b$ 或 $Dy(a)=b$，其中 y 是微分方程的因变量，*a* 和 *b* 是常数。如果指定的初始条件和边界条件比方程中的因变量个数少，那么所得的解中将包含积分常数 C1、C2 等。

函数 dsolve 的输出结果同 solve 类似，既可以用和因变量个数相同数目的输出参数分别接收每个变量的解，也可以把方程的解写入一个结构数组中。

例 7-34，符号微分方程求解示例。

在命令行窗口中输入：

```
syms x;
f=sym('a*Dx+b*x=0')
f =
    Dx*a + b*x == 0
dsolve(f)
```

输出结果：

```
ans =
    C1*exp(-(b*t)/a)
```

在命令行窗口中输入：

```
dsolve(f,'x(0)=1')
```

输出结果：

```
ans =
    exp(-(b*t)/a)
```

在命令行窗口中输入：

```
dsolve(f,'x(0)=1','m')
```

输出结果：

```
ans =
    exp(-(b*m)/a)
```

在命令行窗口中输入：

```
clear
syms x y;
f1=sym('Dx-3*x+2*y=0')
f1 =
     Dx - 3*x + 2*y == 0
f2=sym('Dy-2*x+y=0')
     f2 =
Dy - 2*x + y == 0
      [x,y]=dsolve(f1,f2)
```

输出结果：

```
x =
      (C2*exp(t))/2 + C1*exp(t) + C2*t*exp(t)
y =
     C1*exp(t) + C2*t*exp(t)
```

在命令行窗口中输入：

```
 [x,y]=dsolve(f1,f2,'x(0)=-1','y(0)=1')
```

输出结果：

```
x =
     - exp(t) - 4*t*exp(t)
y =
     exp(t) - 4*t*exp(t)
```

提示

dsolve 并不总能得到显式解，如果不能得到显式解，dsolve 会尝试求隐式解。如果也不能够得到隐式解，则应当采用 ODE 来求微分方程的数值解。

7.10　符号分析可视化

MATLAB 的符号数学工具箱为符号函数可视化提供了一组简便易用的指令。本节着重介绍两个进行数学分析的可视化界面，即图示化符号函数计算器界面（由指令 funtool 引出）和泰勒级数逼近分析界面（由指令 taylortool 引出）。

7.10.1　funtool 分析界面

对于习惯使用计算器或者只做一些简单的符号运算与图形处理的读者，MATLAB 提供的图示化符号函数计算器是一个较好的选择。该计算器功能虽简单，但操作方便，可视性强。

进入 funtool 分析界面的方法是在 MATLAB 命令行窗口中输入 funtool 命令。

在命令行窗口中输入：

```
funtool
```

弹出如图 7-2 所示的 funtool 分析界面。

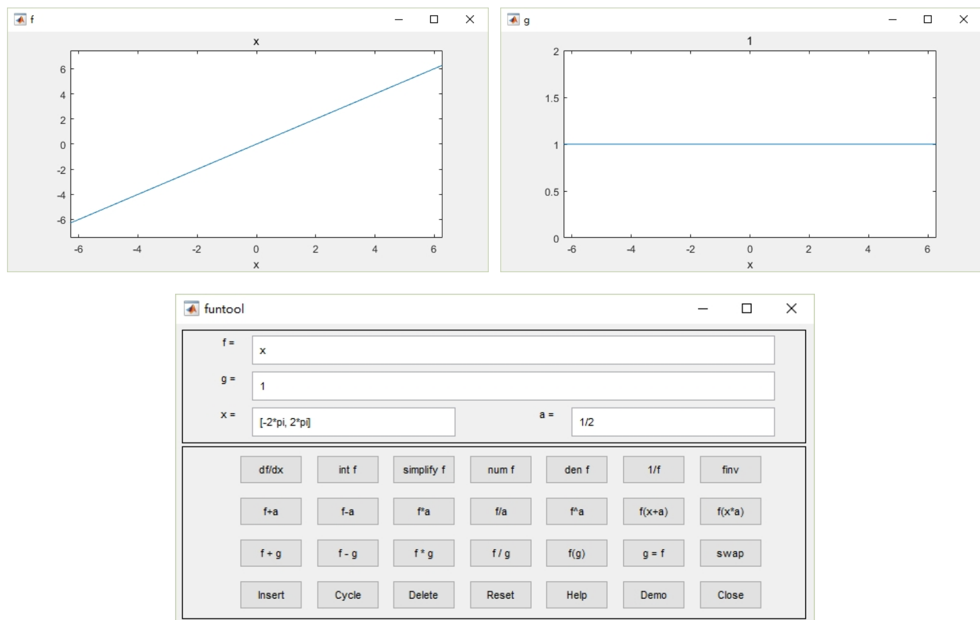


图 7-2　funtool 分析界面

如图 7-2 所示的 funtool 分析界面是由两个图形窗口（f 和 g）与一个函数运算控制窗口（funtool）组成的。在任何时候，两个图形窗口只有一个处于激活状态。函数运算控制窗口上的任何操作都只能对被激活的函数图形窗口起作用，即被激活的函数图像可随运算控制窗口的操作而做相应的变化。

（1）第 1 排按键只对 *f* 起作用，如求导、积分、简化、提取分子和分母、计算 1/*f* 及求反函数。

（2）第 2 排按键处理函数 *f* 和常数 *a* 之间的加、减、乘、除等运算。

（3）第 3 排的前 4 个按键对两个函数 *f* 和 *g* 之间进行算术运算；第 5 个按键求复合函数；第 6 个按键的功能是把 *f* 函数传递给 *g*；最后一个按键 swap 用于实现 *f* 和 *g* 的互换。

（4）第 4 排按键用于对计算器自身进行操作。funtool 计算器有一张函数列表 fxlist。这 7 个按键的功能依次如下：

- Insert：把当前激活窗的函数写入列表。
- Cycle：依次循环显示 fxlist 中的函数。
- Delete：从 fxlist 列表中删除激活窗的函数。
- Reset：使计算器恢复到初始调用状态。
- Help：获得关于界面的在线提示说明。
- Demo：自动演示。
- Close：关闭对话框。

例 7-35，funtool 符号函数运算示例。

在命令行窗口中输入：

```
funtool
```

在 funtool 窗口的 *f* 函数右侧的文本输入框中输入 sin(2**x*)，按 Enter 键，完成 *f* 函数的设置。*f* 函数的图形如图 7-3 所示。

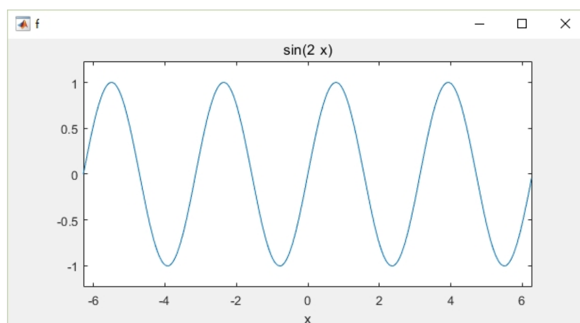


图 7-3　sin(2x)函数图像

在 funtool 窗口的 *g* 函数右侧的文本输入框中输入 cos(*x*)+sin(*x*) 完成 *g* 函数的设置。*g* 函数的图形如图 7-4 所示。

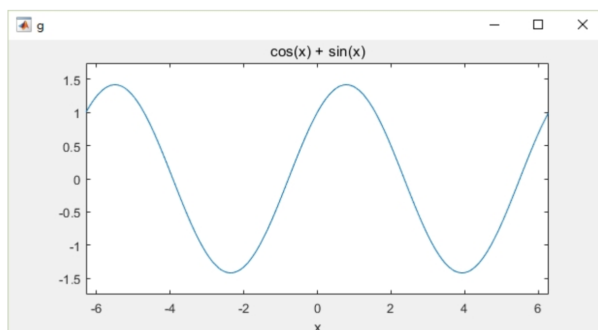


图 7-4　cos(x)+sin(x)函数图像

设置完成的 funtool 如图 7-5 所示。

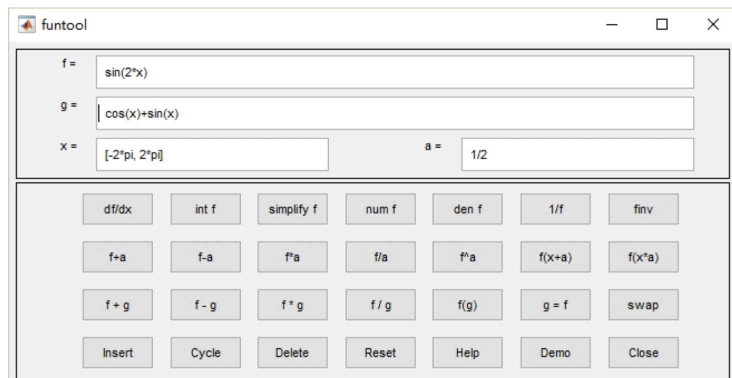


图 7-5　funtool 分析界面的参数设置

下面计算 *f* 函数与 *g* 函数的乘积，单击 funtool 中第 3 排第 3 个按钮，完成函数的乘积，乘积的函数图形显示在 f 窗口中，如图 7-6 所示。

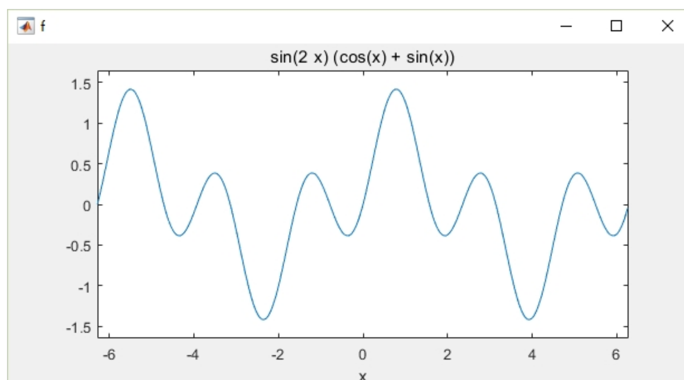


图 7-6　sin(2*x*)(cos(*x*)+sin(*x*))函数图像

计算 *f* 函数与 *g* 函数的复合函数。单击 funtool 中第 3 排第 5 个按钮，完成复合函数 *f*(*g*)，复合函数图形显示在 f 窗口中，如图 7-7 所示。

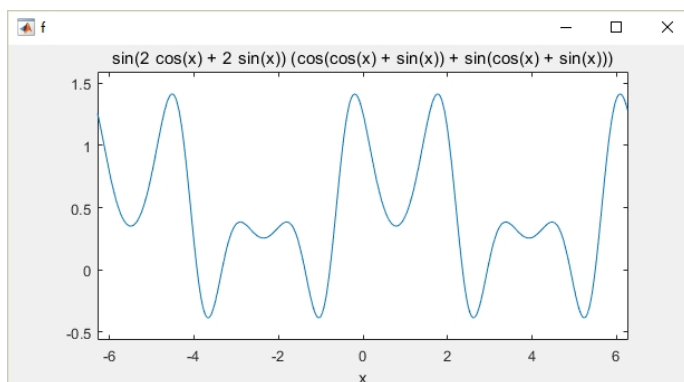


图 7-7　sin(2cos(*x*)+2sin(*x*))复合函数图像

7.10.2　taylortool 分析界面

在命令行窗口中输入：

```
taylortool
```

弹出如图 7-8 所示的泰勒逼近分析界面。

该界面用于观察函数 *f*(*x*)在给定区间上被 *N* 阶泰勒多项式 $T_N(x)$逼近的情况。

函数 *f*(*x*)的输入方式有两种：直接由指令 taylortool(fx)引入；或者在界面的 *f*(*x*)栏中直接输入表达式，然后按 Enter 键输入。

界面中 *N* 被默认设置为 7，可以用其右侧的按钮改变阶次，也可以直接写入阶次。

界面上的 *a* 是级数的展开点，默认值为 0。

函数的观察区被默认设置为(-2π,2π)。

例 7-36，taylortool 符号函数示例。

在命令行窗口中输入：

```
taylortool
```

在 Taylor Tool 图形框 $f(x)$右侧的文本框中输入 1+exp(x)，N 的值采用默认值 7，求解函数 $1+e^x$ 的泰勒展开。展开以后的图形如图 7-9 所示。

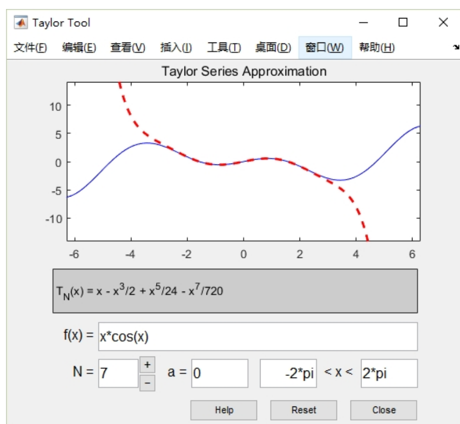


图 7-8　taylortool 分析界面

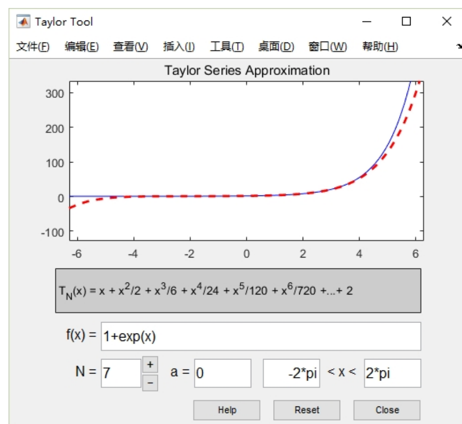


图 7-9　$1+e^x$ 在 0 点的泰勒展开

在图形区可以看到函数的泰勒展开为：

$$T_N(x)=x+x^2/2+x^3/6+x^4/24+x^5/120+x^6/720+\cdots+2$$

下面来对函数 $1+e^x$ 进行 4 次方展开，并且在 1 处对函数进行 4 次方展开，展开的结果分别如图 7-10 和图 7-11 所示。

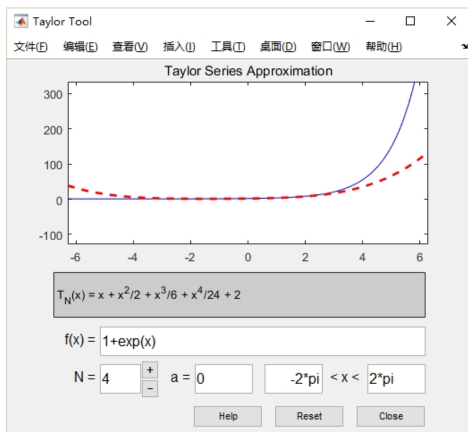


图 7-10　$1+e^x$ 在 0 点的 4 次方泰勒展开

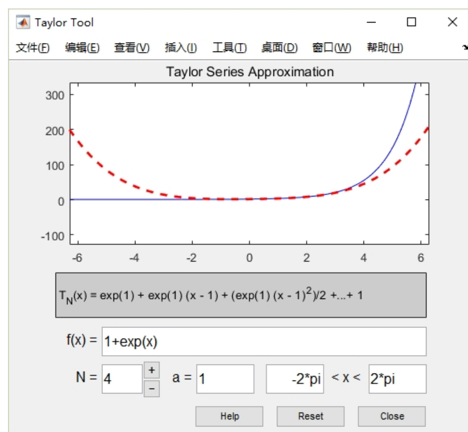


图 7-11　$1+e^x$ 在 1 处的 4 次方泰勒展开

7.11　本章小结

本章介绍了 MATLAB 中的符号计算功能，主要讲解了符号对象及符号表达式的创建、运算及精度；符号表达式的显示、合并、展开、嵌套、分解及化简；符号表达式的替换；符号函数的复合函数和反函数操作；符号表达式的极限、微分、积分、级数求和及泰勒级数展

开；符号函数的傅里叶变换、拉普拉斯变换及 Z 变换；符号代数方程的求解；符号微分方程的求解；符号分析的可视化过程。

通过本章的讲解，能够使读者掌握 MATLAB 所提供的符号计算和分析功能，为后面的学习打下基础，方便读者学习和理解 MATLAB 在数值计算及信号分析领域的应用。

第 8 章

概率统计

知识要点

概率统计是研究自然界中概率统计的方法，是 MATLAB 数据处理的一项重要应用。进行 MATLAB 概率统计有关内容的学习，有利于掌握 MATLAB 的基本使用方法，并结合概率统计有关知识对 MATLAB 在数据处理方面的应用有深刻的了解。

学习要求	学习目标 / 知识点	了解	理解	应用	实践
	产生随机变量		√		√
	概率密度计算			√	√
	累积概率分布			√	√
	统计特征		√		√
	统计作图		√		√



8.1 产生随机变量

MATLAB 中用于产生随机数的函数很多，基本能满足一般应用情况下的需求。下面简单介绍几种随机数的产生方法。

8.1.1 二项分布的随机数据的产生

使用 binornd 函数可以产生二项分布随机数据，其调用格式如下：

- R = binornd(N,P)。
- R = binornd(N,P,m,n,...)。
- R = binornd(N,P,[m,n,...])。

其中，*N*、*P* 为二项分布参数，返回服从参数为 *N*、*P* 的二项分布的随机数 *R*；*m* 指定随机数的个数，与 *R* 同维数；*m*、*n* 分别表示 *R* 的行数和列数。

例 8-1，二项分布的随机数据的产生示例。

在命令行窗口中输入：

```
n = 10:10:60;
r1 = binornd(n,1./n)
r2 = binornd(n,1./n,[1 6])
r3 = binornd(n,1./n,1,6)
r4 = binornd([n; n],[1./n; 1./n],2,6)
```

依次得到如下结果：

```
r1 =
    3    2    1    2    1    0
r2 =
    1    1    2    0    3    1
r3 =
    0    2    1    2    1    4
r4 =
    1    0    2    2    2    1
    2    1    0    1    2    0
```

8.1.2 正态分布的随机数据的产生

使用 normrnd 函数可以产生参数为 μ、σ 的正态分布的随机数据。其调用格式如下：

- R = normrnd(mu,sigma)。
- R = normrnd(mu,sigma,m,n,...)。
- R = normrnd(mu,sigma,[m,n,...])。

该函数返回均值为 mu，标准差为 sigma 的正态分布随机数据；*m* 指定随机数的个数，*n* 表示 *R* 的列数。

例 8-2，产生正态分布的随机数据示例。

在命令行窗口中输入：

```
n1 = [normrnd(1:6,1./(1:6)); normrnd(1:6,1./(1:6)); normrnd(1:6,1./(1:6));]
n2 = normrnd(0,1,[1 5])
n3 = normrnd(0,1,[5 5])
n4 = normrnd([1 2 3;4 5 6],0.1,2,3)
```

依次得到如下结果：

```
n1 =
    0.5752    0.8646    3.5941    3.8032    5.0750    5.7328
    0.5075    2.4798    2.5960    4.1247    4.7638    6.1079
    1.4936    1.9382    3.5980    3.5983    5.1072    6.2968
n2 =
    0.3108    0.7081   -0.1857    0.8816   -1.7964
n3 =
   -1.1317   -0.6197    0.3054   -0.1841   -0.9795
   -1.0901    0.4418    1.2039    0.6550   -1.5686
   -1.1453   -1.6537    0.6172    2.0343    0.9901
    2.3934   -0.7060   -0.0198   -1.1700   -0.4615
    1.7305   -0.2923    1.2425   -0.2807   -1.7017
n4 =
    0.8976    2.0647    2.8415
    4.0986    5.0113    5.9480
```

8.1.3　常见分布的随机数产生

常见分布的随机数产生函数的使用格式与二项分布随机数据和正态分布随机数据的产生函数相同。相关的函数如表 8-1 所示。

表 8-1　随机数产生函数表

函　　数	说　　明
unifrnd	$[a,b]$上均匀分布（连续）随机数
unidrnd	均匀分布（离散）随机数
exprnd	参数为 Lambda 的指数分布随机数
normrnd	参数为 mu、sigma 的正态分布随机数
chi2rnd	自由度为 n 的卡方分布随机数
trnd	自由度为 n 的 t 分布随机数
Frnd	第一自由度为 N_1、第二自由度为 N_2 的 F 分布随机数
gamrnd	参数为 a、b 的 γ 分布随机数
betarnd	参数为 a、b 的 β 分布随机数
lognrnd	参数为 mu、sigma 的对数正态分布随机数
nbinrnd	参数为 r、p 的负二项式分布随机数
ncfrnd	参数为 n_1、n_2、delta 的非中心 F 分布随机数
nctrnd	参数为 n、delta 的非中心 t 分布随机数
ncx2rnd	参数为 n、delta 的非中心卡方分布随机数
raylrnd	参数为 b 的瑞利分布随机数
weibrnd	参数为 a、b 的韦伯分布随机数

（续表）

函　　数	说　　明
binornd	参数为 n、p 的二项分布随机数
geornd	参数为 p 的几何分布随机数
hygernd	参数为 M、K、N 的超几何分布随机数
poissrnd	参数为 Lambda 的泊松分布随机数

8.2　概率密度计算

8.2.1　通用函数概率密度值

使用 pdf 函数可以轻松计算概率密度。其调用格式如下：

- y = pdf(name,X,A)。
- y = pdf(name,X,A,B)。
- y = pdf(name,X,A,B,C)。
- y = pdf(obj,X)。

其中，返回在 $x=X$ 处，参数为 A、B、C 的概率密度值，对于不同的分布，参数个数不同；name 为分布函数名，其取值如表 8-2 所示；obj 为高斯联合分布对象。该函数的使用方式如例 8-3 所示。

表 8-2　常见分布函数表

name 的取值	函数说明
'beta'或'Beta'	Beta 分布
'bino'或'Binomial'	二项分布
'chi2'或'Chisquare'	卡方分布
'exp'或'Exponential'	指数分布
'f'或'F'	F 分布
'gam'或'Gamma'	GAMMA 分布
'geo'或'Geometric'	几何分布
'hyge'或'Hypergeometric'	超几何分布
'logn'或'Lognormal'	对数正态分布
'nbin'或'Negative Binomial'	负二项式分布
'ncf'或'Noncentral F'	非中心 F 分布
'nct'或'Noncentral t'	非中心 t 分布
'ncx2'或'Noncentral Chi-square'	非中心卡方分布
'norm'或'Normal'	正态分布
'poiss'或'Poisson'	泊松分布

（续表）

name 的取值	函数说明
'rayl'或'Rayleigh'	瑞利分布
't'或'T'	T 分布
'unif'或'Uniform'	均匀分布
'unid'或'Discrete Uniform'	离散均匀分布
'weib'或'Weibull'	Weibull 分布

而且，通用函数计算概率密度的方法还可以推广到任意函数/数据的情况。在 MATLAB 中，可以使用 ksdensity 函数求取一般函数/数据的概率密度函数。该函数的调用格式如下：

- [f,xi] = ksdensity(x)。
- f = ksdensity(x,xi)。
- ksdensity(...)。
- ksdensity(ax,...)。
- [f,xi,u] = ksdensity(...)。
- [...] = ksdensity(...,'Name',value)。

其中，***x*** 为待统计的向量；*xi* 计算概率密度的点；*f* 为得到的概率密度；*ax* 指定绘制位置坐标轴对象；Name 和 value 为可选属性及其属性值。该函数的使用方法将结合例 8-3 进行说明。

例 8-3，计算标准正态分布随机变量在[-2:1:2]处的概率密度值、泊松分布随机变量在点[0:1:4]的密度函数值、高斯联合分布的概率密度函数曲面和任意函数/数据的概率密度分布。

在命令行窗口中输入：

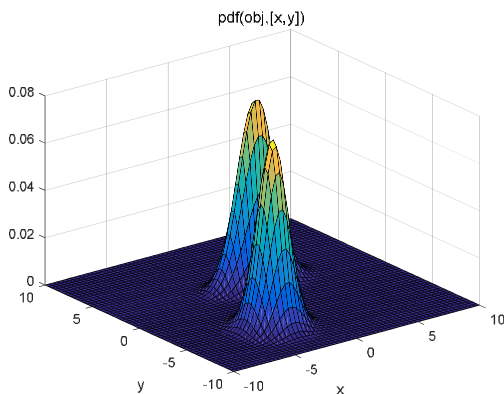
```
p1 = pdf('Normal',-2:2,0,1)
p2 = pdf('Poisson',0:4,1:5)
MU = [1 2;-3 -5];
SIGMA = cat(3,[2 0;0 .5],[1 0;0 1]);
p = ones(1,2)/2;
obj = gmdistribution(MU,SIGMA,p)
ezsurf(@(x,y)pdf(obj,[x y]),[-10 10],[-10 10])
RAND=randn(1000,1);
fx=sin((1:1000)*pi/500);
[f,xi]=ksdensity(RAND+5*fx');
figure;
plot(xi,f);
axis tight
```

输出结果：

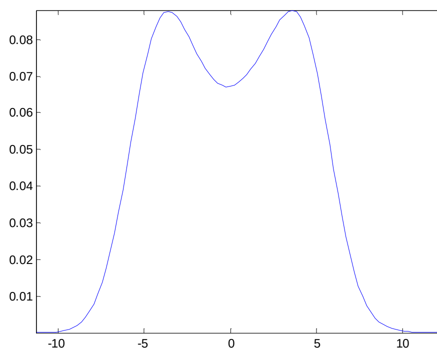
```
p1 =
   0.0540   0.2420   0.3989   0.2420   0.0540
p2 =
   0.3679   0.2707   0.2240   0.1954   0.1755
obj =
Gaussian mixture distribution with 2 components in 2 dimensions
Component 1:
Mixing proportion: 0.500000
Mean:    1    2
```

```
Component 2:
Mixing proportion: 0.500000
Mean:    -3    -5
```

程序运行得到的图形如图 8-1 所示。



（a）高斯联合分布概率密度



（b）任意函数的概率分布示例

图 8-1　概率密度示例

8.2.2　专用函数概率密度值

计算专用函数概率密度值的函数很多。例如，使用 binopdf，其调用格式如下：

- Y = binopdf(X,N,P)：*X* 为计算点；*N* 为试验总次数；*P* 为每次试验中事件发生的概率；*Y* 为概率密度值。

常见的专用概率密度值计算函数如表 8-3 所示。

表 8-3　专用概率密度值计算函数

函数名	调用格式	注　　释
unifpdf	unifpdf (x, a, b)	[*a*,*b*]上均匀分布（连续）概率密度在 *X*=*x* 处的函数值
unidpdf	Unidpdf(x,n)	均匀分布（离散）概率密度函数值
exppdf	exppdf(x, Lambda)	参数为 Lambda 的指数分布概率密度函数值
normpdf	normpdf(x, mu, sigma)	参数为 mu、sigma 的正态分布概率密度函数值
chi2pdf	chi2pdf(x, n)	自由度为 *n* 的卡方分布概率密度函数值
tpdf	tpdf(x, n)	自由度为 *n* 的 t 分布概率密度函数值
fpdf	fpdf(x, n1, n2)	第一自由度为 *n*1、第二自由度为 *n*2 的 F 分布概率密度函数值
gampdf	gampdf(x, a, b)	参数为 *a*、*b* 的 γ 分布概率密度函数值
Betapdf	betapdf(x, a, b)	参数为 *a*、*b* 的 β 分布概率密度函数值
lognpdf	lognpdf(x, mu, sigma)	参数为 mu、sigma 的对数正态分布概率密度函数值
nbinpdf	nbinpdf(x, R, P)	参数为 *R*、*P* 的负二项式分布概率密度函数值
ncfpdf	ncfpdf(x, n1, n2, delta)	参数为 *n*1、*n*2、delta 的非中心 F 分布概率密度函数值

（续表）

函数名	调用格式	注　　释
nctpdf	nctpdf(x, n, delta)	参数为 *n*、delta 的非中心 t 分布概率密度函数值
ncx2pdf	ncx2pdf(x, n, delta)	参数为 *n*、delta 的非中心卡方分布概率密度函数值
raylpdf	raylpdf(x, b)	参数为 *b* 的瑞利分布概率密度函数值
weibpdf	weibpdf(x, a, b)	参数为 *a*、*b* 的韦伯分布概率密度函数值
binopdf	binopdf(x,n,p)	参数为 *n*、*p* 的二项分布概率密度函数值
geopdf	geopdf(x,p)	参数为 *p* 的几何分布概率密度函数值
hygepdf	hygepdf(x,M,K,N)	参数为 *M*、*K*、*N* 的超几何分布概率密度函数值
poisspdf	poisspdf(x,Lambda)	参数为 Lambda 的泊松分布概率密度函数值

例 8-4，计算正态分布函数概率密度示例。

在命令行窗口中输入：

```
mu = [0:0.1:2];
[y i] = max(normpdf(1.5,mu,1));
MLE = mu(i)
```

在命令行窗口输出：

```
MLE =
  1.5000
```

8.3　累积概率分布

8.3.1　通用函数累积概率值

使用函数 cdf 可以计算随机变量 $x \leqslant X$ 的概率之和（累积概率值）。其调用格式如下：

- Y = cdf('name',X,A)。
- Y = cdf('name',X,A,B)。
- Y = cdf('name',X,A,B,C)。
- Y= cdf(obj,X)。

其中，返回在 $x=X$ 处，参数为 *A*、*B*、*C* 的累积概率值，对于不同的分布，参数个数不同；name 为分布函数名，其取值如表 8-2 所示；obj 为高斯联合分布对象。该函数的使用方式如例 8-5 所示。

而对于任意函数/数据，可以使用 ksdensity 函数来求取函数/数据的累积概率分布。但该函数默认为计算函数/数据的概率密度值，如果要计算累积概率分布，需要设置其属性 function 的取值为 cdf。计算方式可以参考例 8-5。

例 8-5，计算标准正态分布随机变量在[-2:1:2]处的累积概率值、泊松分布随机变量在点[0:1:4]的累积概率值、高斯联合分布的累积概率曲面和任意函数/数据的累积概率分布。

在命令行窗口中输入：

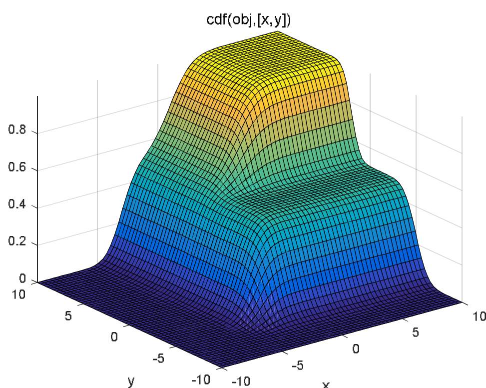
```
p1 = cdf('Normal',-2:2,0,1)
```

```
p2 = cdf('Poisson',0:4,1:5)
MU = [1 2;-3 -5];
SIGMA = cat(3,[2 0;0 .5],[1 0;0 1]);
p = ones(1,2)/2;
obj = gmdistribution(MU,SIGMA,p);
ezsurf(@(x,y)cdf(obj,[x y]),[-10 10],[-10 10])
RAND=randn(1000,1);
fx=sin((1:1000)*pi/500);
[f,xi]=ksdensity(RAND+2000*fx','function','cdf');
figure;
plot(xi,f);
axis tight
```

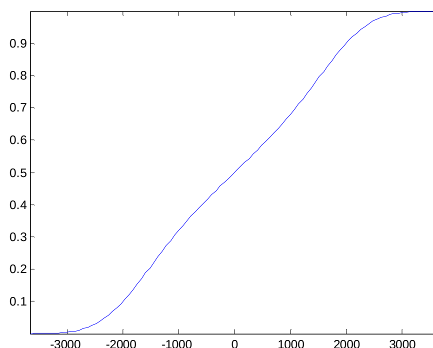
在命令行窗口输出：

```
p1 =
  0.0228  0.1587  0.5000  0.8413  0.9772
p2 =
    0.3679    0.4060    0.4232    0.4335    0.4405
```

输出的图形如图 8-2 所示。



（a）高斯联合分布概率分布



（b）任意函数/数据的累积概率分布示例

图 8-2　概率分布示例

8.3.2　专用函数累积概率值

常见专用函数累积概率值函数如表 8-4 所示。

表 8-4　专用函数累积概率值函数

函数名	调用格式	注　　释
unifcdf	unifcdf (x, a, b)	[*a*,*b*]上均匀分布（连续）累积分布函数值
unidcdf	unidcdf(x,n)	均匀分布（离散）累积分布函数值
expcdf	expcdf(x, Lambda)	参数为 Lambda 的指数分布累积分布函数值
normcdf	normcdf(x, mu, sigma)	参数为 mu、sigma 的正态分布累积分布函数值
chi2cdf	chi2cdf(x, n)	自由度为 *n* 的卡方分布累积分布函数值

（续表）

函数名	调用格式	注　　释
tcdf	tcdf(x, n)	自由度为 *n* 的 t 分布累积分布函数值
fcdf	fcdf(x, n1, n2)	第一自由度为 *n*1、第二自由度为 *n*2 的 F 分布累积分布函数值
gamcdf	gamcdf(x, a, b)	参数为 *a*、*b* 的 γ 分布累积分布函数值
betacdf	betacdf(x, a, b)	参数为 *a*、*b* 的 β 分布累积分布函数值
logncdf	logncdf(x, mu, sigma)	参数为 mu、sigma 的对数正态分布累积分布函数值
nbincdf	nbincdf(x, R, P)	参数为 *R*、*P* 的负二项式分布累积分布函数值
ncfcdf	ncfcdf(x, n1, n2, delta)	参数为 *n*1、*n*2、delta 的非中心 F 分布累积分布函数值
nctcdf	nctcdf(x, n, delta)	参数为 *n*、delta 的非中心 t 分布累积分布函数值
ncx2cdf	ncx2cdf(x, n, delta)	参数为 *n*、delta 的非中心卡方分布累积分布函数值
raylcdf	raylcdf(x, b)	参数为 *b* 的瑞利分布累积分布函数值
weibcdf	weibcdf(x, a, b)	参数为 *a*、*b* 的韦伯分布累积分布函数值
binocdf	binocdf(x,n,p)	参数为 *n*、*p* 的二项分布累积分布函数值
geocdf	geocdf(x,p)	参数为 *p* 的几何分布累积分布函数值
hygecdf	hygecdf(x,M,K,N)	参数为 *M*、*K*、*N* 的超几何分布累积分布函数值
poisscdf	poisscdf(x,Lambda)	参数为 Lambda 的泊松分布累积分布函数值

例 8-6，求解标准整体分布在区间[-1, 1]上的累积概率分布示例。

在命令行窗口中输入：

```
p = normcdf([-1 1]);
p(2)-p(1)
```

输出结果：

```
ans =
 0.6827
```

8.4　统计特征

本节讲解常见的数值统计特征的求取方法。

8.4.1　平均值、中值

使用 mean、median、nanmedian、geomean、harmmean 函数可以分别求取数据的平均值、中位数、忽略 NaN 的中位数、几何平均数及调和平均数。这些函数的使用很简单，本书不再进行讲解，下面以示例说明。

例 8-7，计算矩阵的平均值、中位数、忽略 NaN 的中位数、几何平均数及调和平均数示例。

在命令行窗口中输入：

```
A=magic(5)
M1=mean(A)
```

```
M2=median(A)
M3=nanmedian(A)
M4=geomean(A)
M5=harmmean(A)
```

输出结果：

```
A =
   17   24    1    8   15
   23    5    7   14   16
    4    6   13   20   22
   10   12   19   21    3
   11   18   25    2    9
M1 =
   13   13   13   13   13
M2 =
   11   12   13   14   15
M3 =
   11   12   13   14   15
M4 =
  11.1462  10.9234   8.4557   9.8787  10.7349
M5 =
   9.2045   9.1371   3.8098   6.2969   8.0767
```

8.4.2 数据比较

数据比较是指由数据比较引发的各种数据操作，常见的操作包括普通排序、按行排序和求解值域大小等，可以通过 sort、sortrows 和 range 函数实现。各个函数较为简单，下面以示例说明。

例 8-8，随机矩阵的普通排序、按行排序和求解值域示例。

在命令行窗口中输入以下命令：

```
A=rand(5)
Y1=sort(A)
Y2=sortrows(A)
Y3=range(A)
```

输出结果：

```
A =
   0.1925   0.0427   0.4991   0.8530   0.6403
   0.1231   0.6352   0.5358   0.8739   0.4170
   0.2055   0.2819   0.4452   0.2703   0.2060
   0.1465   0.5386   0.1239   0.2085   0.9479
   0.1891   0.6952   0.4904   0.5650   0.0821
Y1 =
   0.1231   0.0427   0.1239   0.2085   0.0821
   0.1465   0.2819   0.4452   0.2703   0.2060
   0.1891   0.5386   0.4904   0.5650   0.4170
   0.1925   0.6352   0.4991   0.8530   0.6403
   0.2055   0.6952   0.5358   0.8739   0.9479
Y2 =
   0.1231   0.6352   0.5358   0.8739   0.4170
   0.1465   0.5386   0.1239   0.2085   0.9479
   0.1891   0.6952   0.4904   0.5650   0.0821
```

```
   0.1925   0.0427   0.4991   0.8530   0.6403
   0.2055   0.2819   0.4452   0.2703   0.2060
Y3 =
   0.0824   0.6525   0.4119   0.6655   0.8659
```

8.4.3　期望

期望即为平均值，其计算方式见 8.4.1 节。

8.4.4　方差和标准差

MATLAB 使用 var 和 std 函数分别计算方差和标准差，还可以使用 skewness 函数求解三阶统计量斜度。下面以示例说明这 3 个函数的使用方法。

例 8-9，求解随机数矩阵的方差、标准差和斜度示例。

在命令行窗口中输入：

```
X=randn(2,8)
DX=var(X')
DX1=var(X',1)
S=std(X',1)
S1=std(X')
SK = skewness(X')
SK1 = skewness(X',1)
```

输出结果：

```
X =
   1.0212   0.4147   0.3493   0.3268  -0.8964   1.0378  -0.1729  -0.2971
  -0.8740   0.3484  -0.7292  -0.5149  -1.2033  -0.8459  -1.2087  -3.2320
DX =
   0.4350   1.0321
DX1 =
   0.3806   0.9031
S =
   0.6169   0.9503
S1 =
   0.6595   1.0159
SK =
   -0.2902  -1.1422
SK1 =
   -0.2902  -1.1422
```

8.4.5　协方差与相关系数

MATLAB 使用 cov 和 corrcoef 函数分别计算数据的协方差和相关系数。

例 8-10，使用 cov 和 corrcoef 函数分别计算数据的协方差和相关系数示例。

在命令行窗口中输入：

```
x=ones(1,5)
r=rand(5,1)
X=ones(5)
```

```
A=magic(5)
C1=cov(x)
C2=cov(r)
C3=cov(x,r)
C4=cov(X)
C5=cov(A)
C6=corrcoef(x,r)
C7=corrcoef(X,A)
C8=corrcoef(A)
```

输出结果：

```
x =
     1     1     1     1     1
r =
    0.0464
    0.5054
    0.7614
    0.6311
    0.0899
X =
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
C1 =
     0
C2 =
    0.1040
C3 =
         0         0
         0    0.1040
C4 =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
C5 =
   52.5000    5.0000  -37.5000  -18.7500   -1.2500
    5.0000   65.0000   -7.5000  -43.7500  -18.7500
  -37.5000   -7.5000   90.0000   -7.5000  -37.5000
  -18.7500  -43.7500   -7.5000   65.0000    5.0000
   -1.2500  -18.7500  -37.5000    5.0000   52.5000
C6 =
   NaN   NaN
   NaN     1
C7 =
   NaN   NaN
   NaN     1
C8 =
```

```
    1.0000    0.0856   -0.5455   -0.3210   -0.0238
    0.0856    1.0000   -0.0981   -0.6731   -0.3210
   -0.5455   -0.0981    1.0000   -0.0981   -0.5455
   -0.3210   -0.6731   -0.0981    1.0000    0.0856
   -0.0238   -0.3210   -0.5455    0.0856    1.0000
```

8.5　统计作图

8.5.1　正整数频率表

使用 tabulate 函数可以得到正整数频率表。

例 8-11，向量的正整数统计频率示例。

在命令行窗口中输入：

```
T=ceil(5*rand(1,10))
table = tabulate(T)
```

输出结果：

```
T =
    4    5    5    1    2    2    4    3    4    2
table =
    1    1   10
    2    3   30
    3    1   10
    4    3   30
    5    2   20
```

8.5.2　累积分布函数图形

使用 cdfplot 函数可以绘制累积分布函数的图形。该函数的调用格式如下：

- cdfplot(x)。
- h = cdfplot(x)。
- [h,stats] = cdfplot(x)。

其中，***x*** 为向量；***h*** 为表示曲线的句柄；stats 表示样本的一些特征。

例 8-12，绘制一个极值分布向量的实际概率分布图形和理论概率分布图形。

在命令行窗口中输入如下程序：

```
y = evrnd(0,3,100,1);
cdfplot(y)
hold on
x = -20:0.1:10;
f = evcdf(x,0,3);
plot(x,f,'m')
legend('Empirical','Theoretical','Location','NW')
```

程序运行结果如图 8-3 所示。

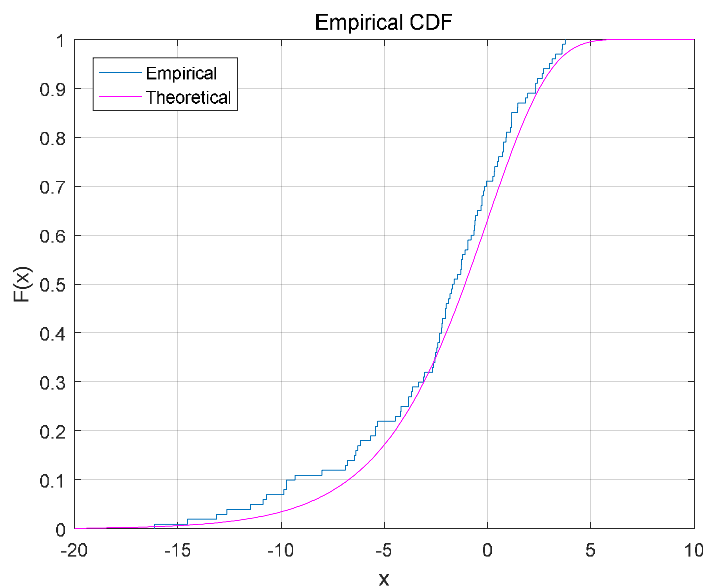


图 8-3　累积分布函数图形

8.5.3　最小二乘拟合直线

使用 lsline 函数可以实现离散数据的最小二乘拟合。该函数的调用格式如下：

- lsline。
- h = lsline。

其中，*h* 为拟合曲线的句柄。

例 8-13，使用 lsline 函数实现离散数据的最小二乘拟合示例。

在命令行窗口中输入以下命令：

```
x = 1:10;
y1 = x + randn(1,10);
scatter(x,y1,25,'b','*')
hold on
y2 = 2*x + randn(1,10);
plot(x,y2,'mo')
y3 = 3*x + randn(1,10);
plot(x,y3,'rx:')
y4 = 4*x + randn(1,10);
plot(x,y4,'g+--')
lsline
```

程序运行结果如图 8-4 所示。从图中可以看到，对添加了随机数据的曲线数据，lsline 函数很好地实现了拟合。

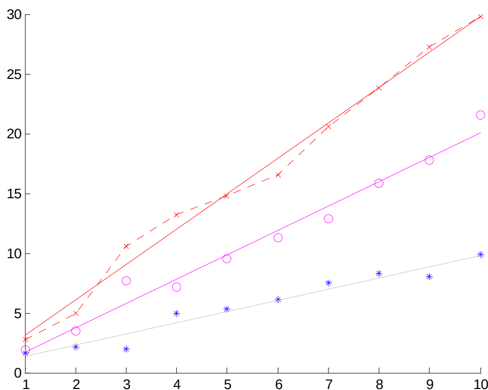


图 8-4　最小二乘拟合直线

8.5.4　绘制正态分布概率图形

使用 normplot 函数可以绘制正态分布概率图形。该函数的调用格式如下：

- h = normplot(X)

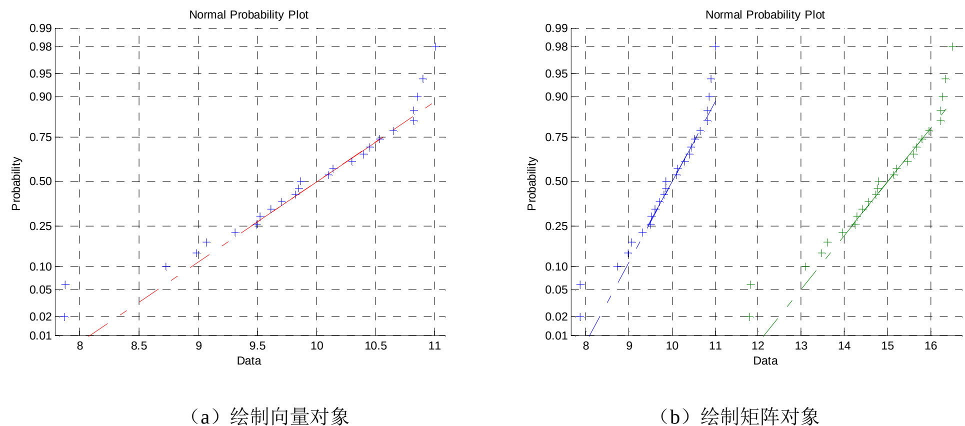
其中，若 ***X*** 为向量，则显示正态分布概率图形；若 ***X*** 为矩阵，则显示每一列的正态分布概率图形。

例 8-14，绘制正态分布概率图形示例。

在命令行窗口中输入下列命令：

```
x = normrnd(10,1,25,1);
normplot(x)
figure;
normplot([x,1.5*x])
```

程序运行结果如图 8-5 所示。



（a）绘制向量对象　　　　　　（b）绘制矩阵对象

图 8-5　绘制正态分布概率图形示意图

8.5.5 样本数据的盒图

使用 boxplot 函数可以绘制样本数据的盒图。该函数的调用格式如下：

- boxplot(X)。
- boxplot(X,G)。
- boxplot(axes,X,...)。
- boxplot(...,'Name',value)。

其中，*X* 为待绘制的变量；*G* 为附加群变量；axes 为坐标轴句柄；Name、value 为可设置属性的属性名和属性值。

例 8-15，样本数据的盒图绘制示例。

在命令行窗口中输入：

```
X = randn(100,25);
subplot(3,1,1)
boxplot(X)
subplot(3,1,2)
boxplot(X,'plotstyle','compact')
subplot(3,1,3)
boxplot(X,'notch','on')
```

程序运行结果如图 8-6 所示。

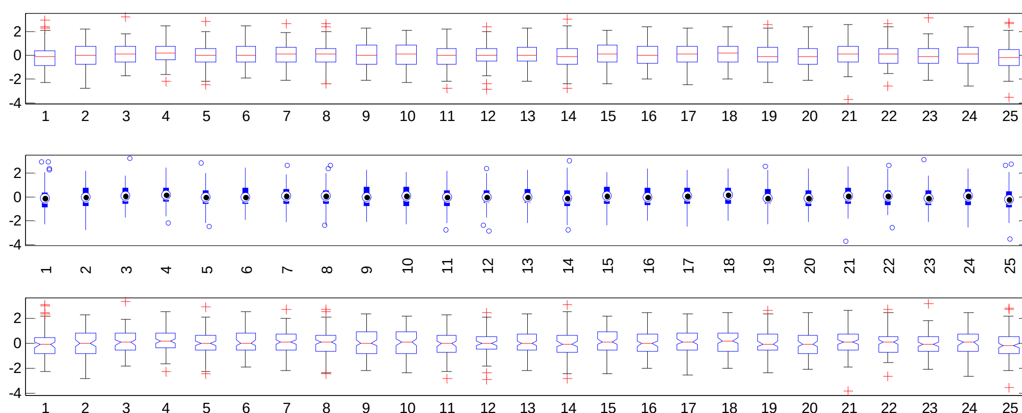


图 8-6 样本数据的盒图绘制示例

8.5.6 参考线绘制

在 MATLAB 中可以使用 refline 和 refcurve 函数分别绘制一条参考直线与一条参考曲线。refline 函数的调用格式如下：

- refline(m,b)。
- refline(coeffs)。
- refline。

- hline = refline(...)。

其中，*m* 为斜率、*b* 为截距；coeffs 为前面两个参数构成的向量；hline 为参考线句柄。

例 8-16，绘制参考直线示例。

在命令行窗口中输入以下命令：

```
x = 1:10;
y = x + randn(1,10);
scatter(x,y,25,'b','*')
lsline
mu = mean(y);
hline = refline([0 mu]);
set(hline,'Color','r')
```

程序运行结果如图 8-7 所示。

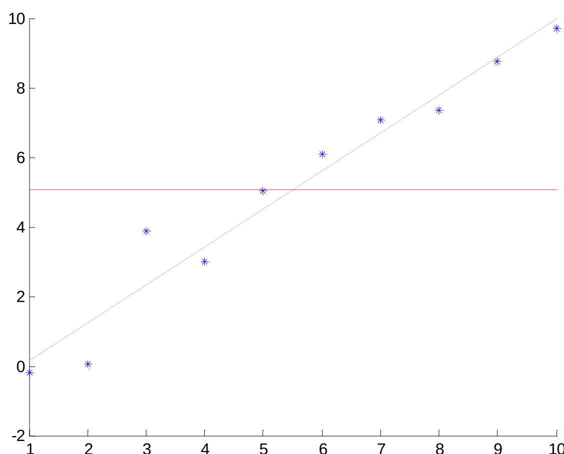


图 8-7　绘制参考直线示例

reflcurve 函数的调用格式如下：

- refcurve(p)。
- refcurve。
- hcurve = refcurve(...)。

其中，***p*** 为多项式系数向量。

例 8-17，绘制参考曲线示例。

在命令行窗口中输入以下命令：

```
p = [1 -2 -1 0];
t = 0:0.1:3;
y = polyval(p,t) + 0.5*randn(size(t));
plot(t,y,'ro')
h = refcurve(p);
set(h,'Color','r')
q = polyfit(t,y,3);
refcurve(q)
legend('Data','Population Mean','Fitted Mean', 'Location','NW')
```

程序运行结果如图 8-8 所示。

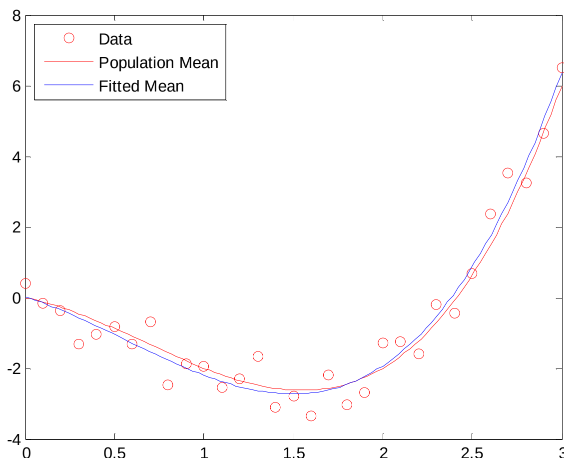


图 8-8　绘制参考曲线示例

8.5.7　样本概率图形

使用 capaplot 函数可以绘制样本的概率图形。函数的调用格式如下：

- p = capaplot(data,specs)。
- [p,h] = capaplot(data,specs)。

其中，data 为所给样本数据，specs 用于指定范围，*p* 表示在指定范围内的概率。该函数返回来自于估计分布的随机变量落在指定范围内的概率。

例 8-18，样本概率图形绘制示例。

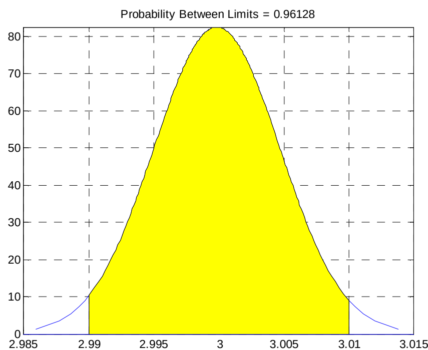
在命令行窗口中输入以下命令：

```
data = normrnd(3,0.005,100,1);
p1=capaplot(data,[2.99 3.01])  %参考图 8-9（a）
grid on; axis tight
figure
p2=capaplot(data,[2.995 3.015])% 参考图 8-9（b）
grid on; axis tight
```

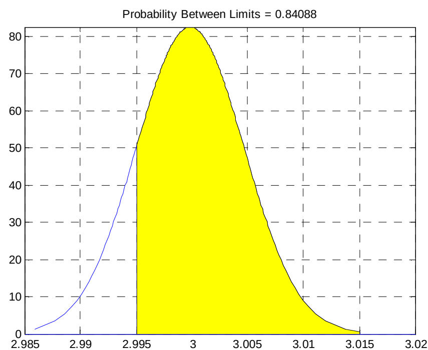
输出结果：

```
p1 =
   0.9613
p2 =
   0.8409
```

输出图形如图 8-9 所示。



（a）区间 1 概率



（b）区间 2 概率

图 8-9　样本概率图形绘制示例

8.5.8　正态拟合直方图

使用 histfit 函数可以绘制含有正态拟合曲线的直方图。该函数的调用格式如下：

- histfit(data)。
- histfit(data,nbins)。
- histfit(data,nbins,dist)。
- h = histfit(...)。

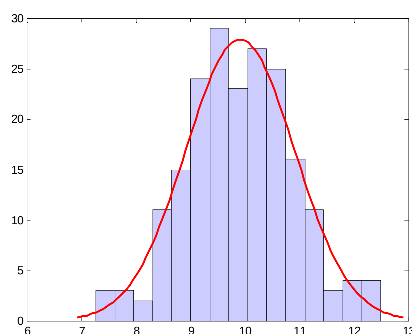
其中，data 为向量；nbins 指定 bar 的个数；dist 为分布类型。

例 8-19，绘制含有正态拟合曲线的直方图示例。

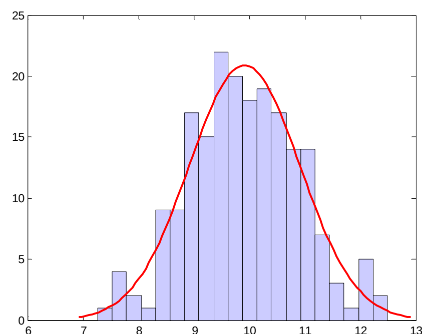
在命令行窗口中输入以下命令：

```
r = normrnd(10,1,200,1);
histfit(r)
h = get(gca,'Children');
set(h(2),'FaceColor',[.8 .8 1])
figure
histfit(r,20)
h = get(gca,'Children');
set(h(2),'FaceColor',[.8 .8 1])
```

程序运行的结果如图 8-10 所示。



（a）柱条数由程序设置



（b）柱条数设置为 20

图 8-10　正态拟合直方图形绘制示例

8.6　本章小结

本章主要介绍了使用 MATLAB 实现概率统计基本数据处理方法相关的基础内容，包括产生随机变量、计算概率密度、计算累积概率分布、挖掘统计特征与统计作图等。

在实际的概率统计应用中，MATLAB 还可以进行更多的数据处理，例如参数估计、假设检验等。因其他内容涉及较深的概率统计知识，本书略去这些内容，如有需要，可以参考 MATLAB 帮助中的相关内容。

第 9 章

数学建模函数

知识要点

本书在前面的几章已经详细介绍了 MATLAB 中的各种基础知识及编程，本章重点介绍常用于 MATLAB 建模的函数，包括曲线拟合函数、参数拟合函数等内容。

学习要求	知识点 \ 学习目标	了解	理解	应用	实践
	曲线拟合函数		√		√
	参数估计函数			√	√
	参数传递			√	√
	全局变量		√		√



9.1 曲线拟合函数

在科学和工程领域，曲线拟合的主要功能是寻求平滑的曲线来最好地表现带有噪声的测量数据，从这些测量数据中寻求两个函数变量之间的关系或者变化趋势，最后得到曲线拟合的函数表达式 $y = f(x)$ 。

使用多项式进行数据拟合会出现数据振荡，而 Spline 插值的方法可以得到很好的平滑效果，但是关于该插值方法有太多的参数，不适合曲线拟合的方法。

同时，由于在进行曲线拟合的时候，已经认为所有测量数据中已经包含噪声，因此，最后的拟合曲线并不要求通过每一个已知数据点，衡量拟合数据的标准则是整体数据拟合的误差最小。

一般情况下，MATLAB 的曲线拟合方法用的是“最小方差”函数，其中方差的数值是拟合曲线和已知数据之间的垂直距离。

9.1.1 多项式拟合

在 MATLAB 中，函数 polyfit()采用最小二乘法对给定的数据进行多项式拟合，得到该多项式的系数。该函数的调用格式如下：

- polyfit(x,y,n)：找到次数为 *n* 的多项式系数，对于数据集合{(x_i, y_i)}，满足差的平方和最小。
- [p,E]=polyfit(x,y,n)：返回同上的多项式 ***P*** 和矩阵 ***E***。多项式系数在向量 ***p*** 中，矩阵 ***E*** 用在 polyval 函数中来计算误差。

例 9-1，某数据的横坐标为 *x*=[0.2 0.3 0.5 0.6 0.8 0.9 1.2 1.3 1.5 1.8]，纵坐标为 *y*=[1 2 3 5 6 7 6 5 4 1]，对该数据进行多项式拟合。

MATLAB 代码：

```
clear all
clc
x=[0.3 0.4 0.7 0.9 1.2 1.9 2.8 3.2 3.7 4.5];
y=[1 2 3 4 5 2 6 9 2 7];
p5=polyfit(x,y,5);                  %5 阶多项式拟合
y5=polyval(p5,x);
p5=vpa(poly2sym(p5),5)              %显示 5 阶多项式
p9=polyfit(x,y,9);                  %9 阶多项式拟合
y9=polyval(p9,x);
figure;                             %画图显示
plot(x,y,'bo');
hold on;
plot(x,y5,'r:');
plot(x,y9,'g--');
legend('原始数据','5 阶多项式拟合','9 阶多项式拟合');
xlabel('x');
ylabel('y');
```

运行程序后，得到的 5 阶多项式如下：

```
p5 =
   0.8877*x^5 - 10.3*x^4 + 42.942*x^3 - 77.932*x^2 + 59.833*x - 11.673
```

运行程序后，得到的输出结果如图 9-1 所示。由图可以看出，使用 5 次多项式拟合时，得到的结果比较差。

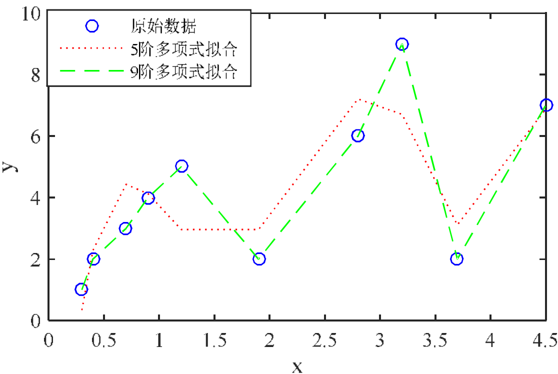


图 9-1　多项式曲线拟合结果

当采用 9 次多项式拟合时，得到的结果与原始数据符合得比较好。当使用函数 polyfit()进行拟合时，多项式的阶次最大不超过 length(x)-1。

9.1.2　加权最小方差（WLS）拟合原理及实例

所谓加权最小方差，就是根据基础数据本身各自的准确度的不同，在拟合的时候给每个数据以不同的加权数值。这种方法比前面介绍的单纯最小方差方法要更加符合拟合的初衷。

对应 N 阶多项式的拟合公式，所需要求解的拟合系数需要求解线性方程组，其中线性方程组的系数矩阵和需要求解的拟合系数矩阵分别为：

$$A=\begin{pmatrix} x_1^N & \cdots & x_1 & 1 \\ x_2^N & \cdots & x_2 & 1 \\ \vdots & \ddots & \vdots & \\ x_m^N & \cdots & x_m & 1 \end{pmatrix},\quad \theta=\begin{pmatrix} \theta_n \\ \theta_{n-1} \\ \vdots \\ \theta_1 \end{pmatrix}$$

使用加权最小方差方法求解得到拟合系数：

$$\theta_m^n=\begin{pmatrix} \theta^n_{mn} \\ \theta^n_{mn-1} \\ \vdots \\ \theta^n_1 \end{pmatrix}=[A^T MA]^{-1}A^T My$$

其对应的加权最小方差为表达式

$$J_m=[A\theta-y]^T W[A\theta-y]$$

例 9-2，根据 WLS 数据拟合方法，自行编写使用 WLS 方法拟合数据的 M 函数，然后使

用 WLS 方法进行数据拟合。

在 M 文件编辑器中输入下面的程序代码：

```
function  [th,err,yi]=polyfits(x,y,N,xi,r)
%x,y:数据点系列
%N ： 多项式拟合的系统
%r  ： 加权系数的逆矩阵

M=length(x);
x=x(:);
y=y(:);

%判断调用函数的格式
if nargin==4
%当调用函数的格式为（x,y,N,r）
   if length(xi)==M
         r=xi;
         xi=x;
%当调用函数的格式为(x,y,N,xi)
     else r=1;
     end
%当调用格式为(x,y,N)
elseif nargin==3
        xi=x;
        r=1;
end
%求解系数矩阵
A(:,N+1)=ones(M,1);
for n=N:-1:1
     A(:,n)=A(:,n+1).*x;
end
if length(r)==M
      for m=1:M
           A(m,:)=A(m,:)/r(m);
           y(m)=y(m)/r(m);
      end
end
  %计算拟合系数
th=(A\y)';
ye=polyval(th,x);
err=norm(y-ye)/norm(y);
yi=polyval(th,xi);
```

将上面的代码保存为“polyfits.m”文件。

使用上面的程序代码，对基础数据进行 LS 多项式拟合。在 MATLAB 的命令行窗口中输入下面的程序代码：

```
clear all
clc
x=[-3:1:3]';
y=[1.1650  0.0751  -0.6965  0.0591 0.6268  0.3516  1.6961]';
[x,i]=sort(x);
y=y(i);
xi=min(x)+[0:100]/100*(max(x)-min(x));
for i=1:4
   N=2*i-1;
```

```
        [th,err,yi]=polyfits(x,y,N,xi);
        subplot(2,2,i)
        plot(x,y,'o')
        hold on
        plot(xi,yi,'-')
        grid on
    end
```

得到的拟合结果如图 9-2 所示。

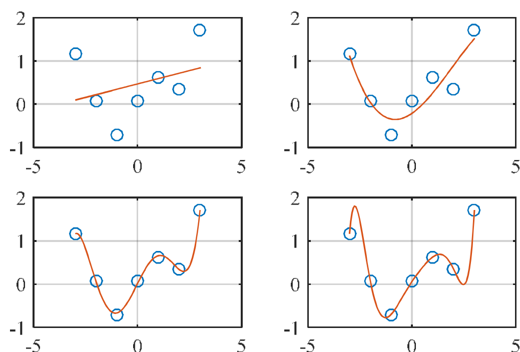


图 9-2　使用 LS 方法求解的拟合结果

从上面的例子可以看出，LS 方法其实是 WLS 方法的一种特例，相当于将每个基础数据的准确度都设为 1。但是，自行编写的 M 文件和默认的命令结果不同，请仔细比较。

9.1.3　非线性曲线拟合

非线性曲线拟合是已知输入向量 *xdata*、输出向量 *ydata*，并知道输入与输出的函数关系为 $ydata=F(x, xdata)$，但不清楚系数向量 *x*。进行曲线拟合即求 *x* 使得下式成立：

$$\min_{x} \frac{1}{2}\left\| F(x, xdata) - ydata \right\|_2^2 = \frac{1}{2}\sum_i \left(F(x, xdata_i) - ydata_i\right)^2$$

在 MATLAB 中，可以使用函数 curvefit 解决此类问题，其调用格式如下：

- x = lsqcurvefit(fun,x0,xdata,ydata)：*x*0 为初始解向量；*xdata*，*ydata* 为满足关系 ydata=F(x, xdata)的数据。
- x = lsqcurvefit(fun,x0,xdata,ydata,lb,ub)：*lb*、*ub* 为解向量的下界和上界，$lb \le x \le ub$，若没有指定界，则 *lb*=[ ]，*ub*=[ ]。
- x = lsqcurvefit(fun,x0,xdata,ydata,lb,ub,options)：options 为指定的优化参数。
- [x,resnorm] = lsqcurvefit(…)：resnorm 是在 *x* 处残差的平方和。
- [x,resnorm,residual] = lsqcurvefit(…)：residual 为在 *x* 处的残差。
- [x,resnorm,residual,exitflag] = lsqcurvefit(…)：exitflag 为终止迭代的条件。
- [x,resnorm,residual,exitflag,output] = lsqcurvefit(…)：output 为输出的优化信息。

例 9-3，已知输入向量 *xdata* 和输出向量 *ydata*，且长度都是 *n*，使用最小二乘非线性拟合函数：

$$ydata(i)=x(1)\cdot xdata(i)^2+x(2)\cdot\sin(xdata(i))+x(3)\cdot xdata(i)^3$$

根据题意可知，目标函数为

$$\min_{x}\ \frac{1}{2}\sum_{i=1}^{n}(F(x,xdata_i)-ydata_i)^2$$

其中：

$$F(x,xdata)=x(1)\cdot xdata^2+x(2)\cdot\sin(xdata)+x(3)\cdot xdata^3$$

初始解向量定位 x0=[0.3, 0.4, 0.1]。

首先建立拟合函数文件 ex1024.m。

```
function F = ex1024 (x,xdata)
F = x(1)*xdata.^2 + x(2)*sin(xdata) + x(3)*xdata.^3;
```

再编写函数拟合代码：

```
clear all
clc
xdata = [3.6 7.7 9.3 4.1 8.6 2.8 1.3 7.9 10.0 5.4];
ydata = [16.5 150.6 263.1 24.7 208.5 9.9 2.7 163.9 325.0 54.3];
x0 = [10, 10, 10];
[x,resnorm] = lsqcurvefit(@ex1024,x0,xdata,ydata)
```

结果为：

```
x =
   0.2269   0.3385   0.3022
resnorm =
   6.2950
```

即函数在 x=0.2269、x=0.3385、x=0.3022 处残差的平方和均为 6.295。

9.2 参数估计函数

参数估计的内容包括点估计和区间估计。MATLAB 统计工具箱提供了很多与参数估计相关的函数，例如计算待估参数及其置信区间、估计服从不同分布的函数的参数。

9.2.1 常见分布的参数估计

MATLAB 统计工具箱提供了多种具体函数的参数估计函数，如表 9-1 所示。

例如，利用 normfit 函数可以对正态分布总体进行参数估计。

- [muhat,sigmahat,muci,sigmaci]=normfit(x)：对于给定的正态分布的数据 x，返回参数 μ的估计值 muhat、σ 的估计值 sigmahat、μ的 95%置信区间 muci、σ 的 95%置信区间 sigmaci。
- [muhat,sigmahat,muci,sigmaci]=normfit(x，alpha)：进行参数估计并计算 100(1-alpha)%置信区间。

表 9-1　常见分布的参数估计函数及其调用格式

分　布	调　用　格　式
贝塔分布	phat=betafit(x) [phat,pci]=betafit(x,alpha)
贝塔对数似然函数	logL=betalike(params,data) [logL,info]=betalike(params,data)
二项分布	phat=binofit(x,n) [phat,pci]=binofit(x,n) [phat,pci]=binofit(x,n,alpha)
指数分布	muhat=expfit(x) [muhat,muci]=expfit(x) [muhat,muci]=expfit(x,alpha)
伽马分布	phat=gamfit(x) [phat,pci]=gamfit(x) [phat,pci]=gamfit(x,alpha)
伽马似然函数	logL=gamlike(params,data) [logL,info]=gamlike(params,data)
最大似然估计	phat= mle('dist',data) [phat,pci]= mle('dist',data) [phat,pci]=mle('dist',data,alpha) [phat,pci]= mle('dist',data,alpha,p1)
正态对数似然函数	L=normlike(params,data)
正态分布	[muhat,sigmahat,muci,sigmaci]=normfit(x) [muhat,sigmahat,muci,sigmaci]=normfit(x,alpha)
泊松分布	lambdahat=poissfit(x) [lambdahat,lambdaci]=poissfit(x) [lambdahat,lambdaci]=poissfit(x,alpha)
均匀分布	[ahat,bhat]=unifit(x) [ahat,bhat,aci,bci]=unifit(x) [ahat,bhat,aci,bci]=unifit(x,alpha)
威布尔分布	phat=weibfit(x) [phat,pci]=weibfit(x) [phat,pci]=weibfit(x,alpha)
威布尔对数似然函数	logL=weiblike(params,data) [logL,info]=weiblike(params,data)

例 9-4，观测某型号 20 辆汽车消耗 10L 汽油的行驶里程，具体数据如下所示：

59.6 55.2 56.6 55.8 60.2 57.4 59.8 56.0 55.8 57.4

56.8 54.4 59.0 57.0 56.0 60.0 58.2 59.6 59.2 53.8

假设行驶里程服从正态分布，请用 normfit 函数求解平均行驶里程的 95%置信区间。

根据题意，可以编写如下 MATLAB 代码：

```
clear all
clc
x1=[59.6 55.2 56.6 55.8 60.2 57.4 59.8 56.0 55.8 57.4];
x2=[56.8 54.4 59.0 57.0 56.0 60.0 58.2 59.6 59.2 53.8];
x=[x1 x2]';
a=0.05;
[muhat,sigmahat,muci,sigmaci]=normfit(x,a);
[p,ci]=mle('norm',x,a);
n=numel(x);
format long
muhat
p1=p(1)
sigmahat
sigmahat1=var(x).^0.5
p2=p(2)
muci
ci
sigmaci
muci1=[muhat-tinv(1-a/2,n-1)*sigmahat/sqrt(n),muhat+tinv(1-a/2,n-1)*sigmahat/sqrt(n)]
sigmaci1=[((n-1).*sigmahat.^2/chi2inv(1-a/2,n-1)).^0.5,((n-1).*sigmahat.^2/chi2inv(a/2,n-1)).^0.5]
```

运行结果：

```
muhat =
  57.390000000000001
p1 =
  57.390000000000001
sigmahat =
   1.966535826750873
sigmahat1 =
   1.966535826750873
p2 =
   1.916742027503963
muci =
  56.469632902339683
  58.310367097660318
ci =
  56.469632902339683   1.495531606349597
  58.310367097660318   2.872266449964584
sigmaci =
   1.495531606349597
   2.872266449964584
muci1 =
  56.469632902339683  58.310367097660318
sigmaci1 =
   1.495531606349597   2.872266449964584
```

9.2.2 点估计

点估计是用单个数值作为参数的估计，目前使用较多的方法是最大似然法和矩法。

1. 最大似然法

最大似然法是在待估参数的可能取值范围内，挑选使似然函数值最大的那个参数值为最大似然估计量。由于最大似然估计法得到的估计量通常不仅仅满足无偏性、有效性等基本条

件，还能保证其为充分统计量，所以，在点估计和区间估计中，一般推荐使用最大似然法。

MATLAB 用函数 mle 进行最大似然估计，其调用格式如下：

- phat=mle('dist',data)：使用 data 矢量中的样本数据，返回 dist 指定的分布的最大似然估计。

例 9-5，观测某型号 20 辆汽车消耗 10L 汽油的行驶里程，具体数据如下所示：

59.6 55.2 56.6 55.8 60.2 57.4 59.8 56.0 55.8 57.4

56.8 54.4 59.0 57.0 56.0 60.0 58.2 59.6 59.2 53.8

假设行驶里程服从正态分布，请用最大似然估计法估计总体的均值和方差。

根据题意，编写最大似然估计求解程序：

```
clear all
clc
x1=[59.6 55.2 56.6 55.8 60.2 57.4 59.8 56.0 55.8 57.4];
x2=[56.8 54.4 59.0 57.0 56.0 60.0 58.2 59.6 59.2 53.8];
x=[x1 x2]';
p=mle('norm',x);
muhatmle=p(1)
sigma2hatmle=p(2)^2
```

运行结果：

```
muhatmle =
  57.390000000000001
sigma2hatmle =
   3.673900000000002
```

2. 矩法

待估参数经常作为总体原点矩或原点矩的函数，此时可以用该总体样本的原点矩或样本原点矩的函数值作为待估参数的估计，这种方法称为矩法。

例如，样本均值总是总体均值的矩估计量，样本方差总是总体方差的矩估计量，样本标准差总是总体标准差的矩估计量。

MATLAB 计算矩的函数为 moment(X,order)。

例 9-6，观测某型号 20 辆汽车消耗 10L 汽油的行驶里程，具体数据如下所示：

59.6 55.2 56.6 55.8 60.2 57.4 59.8 56.0 55.8 57.4

56.8 54.4 59.0 57.0 56.0 60.0 58.2 59.6 59.2 53.8

试估计总体的均值和方差。

根据题意，编写求解程序：

```
clear all
clc
x1=[59.6 55.2 56.6 55.8 60.2 57.4 59.8 56.0 55.8 57.4];
x2=[56.8 54.4 59.0 57.0 56.0 60.0 58.2 59.6 59.2 53.8];
x=[x1 x2]';
muhat=mean(x)
sigma2hat=moment(x,2)
var(x,1)
```

运行结果：

```
muhat =
```

```
  57.390000000000001
sigma2hat =
   3.673900000000002
ans =
   3.673900000000002
```

9.2.3 区间估计

求参数的区间估计，首先要求出该参数的点估计，然后构造一个含有该参数的随机变量，并根据一定的置信水平求该估计值的范围。

在 MATLAB 中用 mle 函数进行最大似然估计时，有如下几种调用格式：

- [phat,pci]=mle('dist',data)：返回最大似然估计和 95%置信区间。
- [phat,pci]=mle('dist',data,alpha)：返回指定分布的最大似然估计值和 100(1- alpha)%置信区间。
- [phat,pci]= mle('dist',data,alpha,p1)：该形式仅用于二项分布，其中 p1 为实验次数。

例 9-7，观测某型号 20 辆汽车消耗 10L 汽油的行驶里程，具体数据如下所示：

59.6 55.2 56.6 55.8 60.2 57.4 59.8 56.0 55.8 57.4

56.8 54.4 59.0 57.0 56.0 60.0 58.2 59.6 59.2 53.8

假设行驶里程服从正态分布，求平均行驶里程的 95%置信区间。

根据题意，编写求解程序：

```
clear all
clc
x1=[29.8 27.6 28.3 27.9 30.1 28.7 29.9 28.0 27.9 28.7];
x2=[28.4 27.2 29.5 28.5 28.0 30.0 29.1 29.8 29.6 26.9];
x=[x1 x2]';
[p,pci]=mle('norm',x,0.05)
```

运行结果：

```
p =
  28.695000000000000   0.958371013751981
pci =
  28.234816451169841   0.747765803174798
  29.155183548830159   1.436133224982292
```

9.3 参数传递

MATLAB 中参数传递过程是按值传递。也就是说，在函数调用过程中，MATLAB 将传入的实际变量值赋值为形式参数指定的变量名，这些变量都存储在函数的变量空间中，这和工作区变量空间是独立的，每一个函数在调用中都有自己独立的函数空间。

例如，在 MATLAB 中编写函数：

```
function y=myfun(x,y)
```

在命令行窗口通过 a=myfun(3,2)调用此函数，那么 MATLAB 首先会建立 myfun 函数的

变量空间，把 3 赋值给 x，把 2 赋值给 y，然后执行函数实现的代码，在执行完毕后，把 myfun 函数返回的参数 y 的值传递给工作区变量 a，调用过程结束后，函数变量空间被清除。

9.3.1　输入和输出参数的数目

MATLAB 的函数可以具有多个输入或输出参数。通常在调用时，需要给出和函数声明语句中一一对应的输入参数；而输出参数个数可以按参数列表对应指定，也可以不指定。不指定输出参数调用函数时，MATLAB 默认把输出参数列表中的第一个参数的数值返回给工作区变量 ans。

MATLAB 中可以通过 nargin 和 nargout 函数，确定函数调用时实际传递的输入和输出参数个数，结合条件分支语句，就可以处理函数调用中指定不同数目的输入输出参数的情况。

例 9-8，输入和输出参数数目的使用。

在命令行窗口中输入：

```
function [n1,n2]=mythe(m1,m2)
if nargin==1
   n1=m1;
   if nargout==2
      n2=m1;
   end
else
   if nargout==1
      n1=m1+m2;
   else
      n1=m1;
      n2=m2;
   end
end
```

运行结果：

```
>> m=mythe(4)
m =
    4
>> [m,n]=mythe(4)
m =
    4
n =
    4
>> m=mythe(4,8)
m =
   12
>> [m,n]=mythe(4,8)
m =
    4
n =
    8
>> mythe(4,8)
ans =
    4
```

指定输入和输出参数个数的情况比较好理解，只要对应函数 M 文件中对应的 if 分支项即

可；而不指定输出参数个数的调用情况，MATLAB 是按照指定了所有输出参数的调用格式对函数进行调用的，不过在输出时只是把第一个输出参数对应的变量值赋给工作区变量 ans。

9.3.2　可变数目的参数传递

函数 nargin 和 nargout 结合条件分支语句，可以处理可能具有不同数目的输入和输出参数的函数调用，但这要求对每一种输入参数数目和输出参数数目的结果分别进行代码编写。

有些情况下，用户可能并不能确定具体调用中传递的输入参数或输出参数的个数，即具有可变数目的传递参数，MATLAB 中可通过 varargin 和 varargout 函数实现可变数目的参数传递，使用这两个函数对于处理具有复杂的输入输出参数个数组合的情况也是便利的。

函数 varargin 和 varargout 把实际的函数调用时传递的参数值封装成一个元胞数组，因此，在函数实现部分的代码编写中，就要用访问元胞数组的方法访问封装在 varargin 和 varargout 中的元胞或元胞内的变量。

例 9-9，可变数目的参数传递。

在命令行窗口中输入：

```
function y=myth(x)
a=0;
for i=1:1:length(x)
   a=a+mean(x(i));
end
y=a/length(x);
```

函数 myth 以 *x* 作为输入参数，从而可以接收可变数目的输入参数，函数实现部分首先计算了各个输入参数（可能是标量、一维数组或二维数组）的均值，然后计算这些均值的均值，运行结果如下所示：

```
>> myth([4 3 4 5 1])
ans =
   3.4000
>> myth(4)
ans =
    4
>> myth([2 3;8 5])
ans =
    5
>> myth(magic(4))
ans =
   8.5000
```

9.3.3　返回被修改的输入参数

前面已经讲过，MATLAB 函数有独立于 MATLAB 工作区的自己的变量空间，因此输入参数在函数内部的修改，都只具有和函数变量空间相同的生命周期，如果不指定将此修改后的输入参数值返回到工作区间，那么在函数调用结束后，这些修改后的值将被自动清除。

例 9-10，函数内部的输入参数修改。

在命令行窗口中输入：

```
function y=mythe(x)
x=x+2;
y=x.^2;
```

在 mythe 函数的内部，首先修改了输入参数 *x* 的值（*x*=*x*+2），然后以修改后的 *x* 值计算输出参数 *y* 的值（y=x.^2）。运行结果如下所示：

```
>> x=2
x =
    2
>> y=mythe(x)
y =
   16
>> x
x =
    2
```

由此结果可见，调用结束后，函数变量区中的 *x* 在函数调用中被修改，但此修改只能在函数变量区有效，这并没有影响到 MATLAB 工作区变量空间中变量 *x* 的值，函数调用前后，MATLAB 工作区中的变量 *x* 始终取值为 2。

那么，如果用户希望函数内部对输入参数的修改也对 MATLAB 工作区的变量有效，那么就需要在函数输出参数列表中返回此输入参数。对于例 9-10 中的函数，则需要把函数修改为 function[*y*,*x*]=mythe(*x*)，而在调用时也要通过[*y*,*x*]=mythe(*x*)语句。

例 9-11，将修改后的输入参数返回给 MATLAB 工作区。

在命令行窗口中输入：

```
function [y,x]=mythee(x)
x=x+2;
y=x.^2;
```

运行结果如下所示：

```
>> x=3
x =
    3
>> [y,x]=mythee(x)
y =
   25
x =
    5
>> x
x =
    5
```

通过函数调用后，MATLAB 工作区中变量 *x* 的取值从 3 变为 8，可见通过[*y*,*x*]=mythee(*x*)调用，实现了函数对 MATLAB 工作区变量的修改。

9.3.4　全局变量

通过返回修改后的输入参数，可以实现函数内部对 MATLAB 工作区变量的修改，而另一种殊途同归的方法是使用全局变量，声明全局辨明需要用到的 global 关键词，调用格式为

global variable。

通过全局变量可以实现 MATLAB 工作区变量空间和多个函数的函数空间共享，这样，多个使用全局变量的函数和 MATLAB 工作区共同维护这一全局变量，任何一处对全局变量的修改，都会直接改变此全局变量的取值。

在应用全局变量时，通常在各个函数内部通过 global variable 语句声明，在命令行窗口或脚本 M 文件中也要先通过 global 声明，然后进行赋值。

例 9-12，全局变量的使用。

在命令行窗口中输入：

```
function y=myt(x)
global a;
a=a+9;
y=cos(x);
```

然后在命令行窗口中声明全局变量赋值调用：

```
>> global a
>> a=2
a =
    2
>> myt(pi)
ans =
   -1
>> cos(pi)
ans =
   -1
>> a
a =
   11
```

通过例 9-12 可见，用 global 将 *a* 声明为全局变量后，函数内部对 *a* 的修改也会直接作用到 MATLAB 工作区中。函数调用一次后，*a* 的值从 2 变为 11。

9.4 本章小结

MATLAB 提供了极其丰富的内部函数，使得用户可以通过命令行调用就可以完成很多工作，但是想要更加高效地利用 MATLAB 建模，离不开 MATLAB 编程。

通过本章的学习，读者应该熟悉并掌握 MATLAB 中各种类型的函数，尤其对曲线拟合函数、参数估计函数和插值函数，需要熟练运用。

第 10 章

经典智能算法



知识要点

人工智能学科诞生于 20 世纪 50 年代中期，当时由于计算机的产生与发展，人们开始了真正意义上的人工智能的研究。它在自动推理、认知建模、机器学习、神经元网络、自然语言处理、专家系统、智能机器人等方面的理论和应用上都取得了成果。

本章主要介绍粒子群算法、遗传算法、蚁群算法三种经典智能算法及其 MATLAB 实现方法。

学习要求	学习目标 知识点	了解	理解	应用	实践
	粒子群算法基本原理		√		√
	粒子群算法应用			√	√
	遗传算法基本原理		√		√
	遗传群算法应用			√	√
	蚁群群算法基本原理		√		√
	蚁群群算法应用			√	√



10.1 粒子群算法的 MATLAB 实现

粒子群优化算法（Particle Swarm Optimization），缩写为 PSO，属于进化算法的一种。和模拟退火算法相似，它也是从随机解出发，通过迭代寻找最优解。它也是通过适应度来评价解的品质，但它比遗传算法规则更为简单，它没有遗传算法的“交叉”（Crossover）和“变异”（Mutation）操作，它通过追随当前搜索到的最优值来寻找全局最优。

10.1.1 基本原理

PSO 从这种模型中得到启示并用于解决优化问题。PSO 中，每个优化问题的潜在解都是搜索空间中的一只鸟，称为粒子。所有的粒子都有一个由被优化的函数决定的适值（fitness value），每个粒子还有一个速度决定它们“飞行”的方向和距离。然后粒子们就追随当前的最优粒子在解空间中搜索。

粒子位置的更新方式如图 10-1 所示。

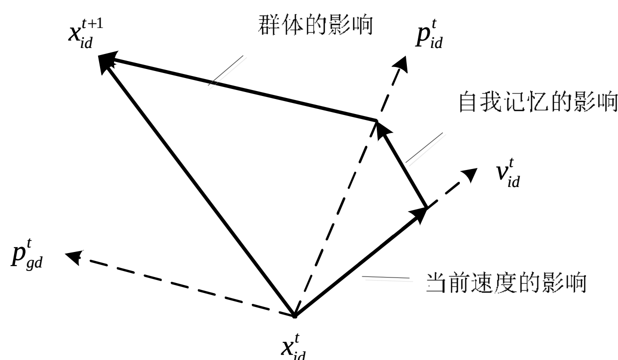


图 10-1 粒子位置的更新方式

其中，x 表示粒子起始位置，v 表示粒子“飞行”的速度，p 表示搜索到的粒子的最优位置。

PSO 初始化为一群随机粒子（随机解），然后通过迭代找到最优解。在每一次迭代中，粒子通过跟踪两个极值来更新自己：第一个就是粒子本身所找到的最优解，这个解称为个体极值；另一个极值是整个种群目前找到的最优解，这个极值是全局极值。另外，也可以不用整个种群而只是用其中一部分作为粒子的邻居，那么在所有邻居中的极值就是局部极值。

假设在一个 D 维的目标搜索空间中，有 N 个粒子组成一个群落，其中第 i 个粒子表示为一个 D 维的向量

$$X_i = (x_{i1}, x_{i2}, \mathbf{L}, x_{iD}), \quad i = 1,2,\mathbf{L},N 。$$

第 i 个粒子的“飞行”速度也是一个 D 维的向量，记为

$$V_i = (v_{i1}, v_{i2}, \mathbf{L}, v_{iD}), \quad i = 1, 2, \mathbf{L}\, 3。$$

第 i 个粒子迄今为止搜索到的最优位置称为个体极值，记为

$$p_{best} = (p_{i1}, p_{i2}, \mathbf{L}, p_{iD}), \quad i = 1, 2, \mathbf{L}, N。$$

整个粒子群迄今为止搜索到的最优位置为全局极值，记为

$$g_{best} = (p_{g1}, p_{g2}, \mathbf{L}, p_{gD})$$

在找到这两个最优值时，粒子根据如下的公式来更新自己的速度和位置：

$$v_{id} = w * v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id})$$

$$x_{id} = x_{id} + v_{id}$$

其中：c_1 和 c_2 为学习因子，也称加速常数（acceleration constant），r_1 和 r_2 为[0，1]范围内的均匀随机数。

式 $v_{id} = w * v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id})$ 右边由三部分组成：

- 第一部分为“惯性”（inertia）或“动量”（momentum）部分，反映了粒子的运动“习惯（habit）”，代表粒子有维持自己先前速度的趋势。
- 第二部分为“认知”（cognition）部分，反映了粒子对自身历史经验的记忆或回忆，代表粒子有向自身历史最佳位置逼近的趋势。
- 第三部分为“社会”（social）部分，反映了粒子间协同合作与知识共享的群体历史经验，代表粒子有向群体或邻域历史最佳位置逼近的趋势。

由于粒子群算法具有高效的搜索能力，有利于得到多目标意义下的最优解；通过代表整个解集种群，按并行方式同时搜索多个非劣解，即搜索到多个 Pareto 最优解。

同时，粒子群算法的通用性比较好，适合处理多种类型的目标函数和约束，并且容易与传统的优化方法结合，从而改进自身的局限性，更高效地解决问题。因此，将粒子群算法应用于解决多目标优化问题上具有很大的优势。

10.1.2 程序设计

基本粒子群算法的流程图，如图 10-2 所示。其具体过程如下：

①初始化粒子群，包括群体规模 N，每个粒子的位置 x_i 和速度 v_i。

②计算每个粒子的适应度值 $F_{it}[i]$。

③对每个粒子，用它的适应度值 $F_{it}[i]$ 和个体极值 $p_{best}(i)$ 比较，如果 $F_{it}[i] > p_{best}(i)$，则用 $F_{it}[i]$ 替换 $p_{best}(i)$。

④对每个粒子，用它的适应度值 $F_{it}[i]$ 和全局极值 g_{best} 比较，如果 $F_{it}[i] > p_{best}(i)$ 则用 $F_{it}[i]$ 替换 g_{best}。

⑤更新粒子的速度 v_i 和位置 x_i。

⑥如果满足结束条件（误差足够好或达到最大循环次数）退出，否则返回②。

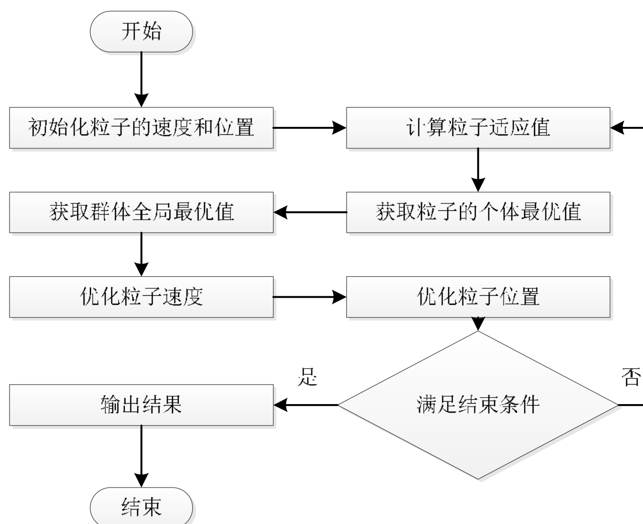


图 10-2　PSO 算法流程图

在 MATLAB 中编程实现的基本粒子群算法基本函数为 PSO。其调用格式如下所示：

```
[xm,fv]=PSO(fitness,N,c1,c2,w,M,D)
```

其中，fitness 为待优化的目标函数，也称适应度函数。N 是粒子数目，c1 是学习因子 1，c2 是学习因子 2，w 是惯性权重，M 是最大迭代数，D 是自变量的个数，xm 是目标函数取最小值时的自变量，fv 是目标函数的最小值。

使用 MATLAB 实现基本粒子群算法代码：

```
function[xm,fv]=PSO(fitness,N,c1,c2,w,M,D)
%%%%%%给定初始化条件%%%%%%%%%
% c1 学习因子 1
% c2 学习因子 2
% w 惯性权重
% M 最大迭代次数
% D 搜索空间维数
% N 初始化群体个体数目
%%%%%初始化种群的个体(可以在这里限定位置和速度的范围) %%%%%%%%%
```

```
format long;
for i=1:N
   for j=1:D
      x(i,j)=randn;        %随机初始化位置
      v(i,j)=randn;        %随机初始化速度
   end
end
%%%%%%先计算各个粒子的适应度，并初始化 Pi 和 Pg%%%%%%%%%%%%%
for i=1:N
   p(i)=fitness(x(i,:));
   y(i,:)=x(i,:);
end
pg=x(N,:);             %Pg 为全局最优
for i=1:(N-1)
   if fitness(x(i,:)) < fitness(pg)
      pg=x(i,:);
   end
end
%%%%%进入主要循环，按照公式依次迭代，直到满足精度要求%%%%%
for t=1:M
   for i=1:N               %更新速度、位移
      v(i,:)=w*v(i,:)+c1*rand*(y(i,:)-x(i,:))+c2*rand*(pg-x(i,:));
      x(i,:)=x(i,:)+v(i,:);
      if fitness(x(i,:)) < p(i)
         p(i)=fitness(x(i,:));
         y(i,:)=x(i,:);
      end
      if p(i)<fitness(pg)
         pg=y(i,:);
      end
   end
   Pbest(t)=fitness(pg);
end
%%%%%%最后给出计算结果
disp('*************************************************')
disp('目标函数取最小值时的自变量：')
xm=pg'
disp('目标函数的最小值为：')
fv=fitness(pg)
disp('**************************************************')
```

将上面的函数保存到 MATLAB 可搜索路径中，即可调用该函数。再定义不同的目标函数 fitness 和其他输入量，就可以用粒子群算法求解不同问题。

粒子群算法使用的函数有很多种，下面介绍两个常用的适应度函数。

1.Griewank 函数

该函数的 MATLAB 代码如下所示：

```
function y=Griewank(x)
%Griewan 函数
%输入 x,给出相应的 y 值,在 x=(0,0,…,0)处有全局极小点 0.
[row,col]=size(x);
if row>1
   error('输入的参数错误');
end
y1=1/4000*sum(x.^2);
```

```
y2=1;
for h=1:col
   y2=y2*cos(x(h)/sqrt(h));
end
y=y1-y2+1;
y=-y;
```

绘制以上函数图形的 MATLAB 代码：

```
function DrawGriewank()
%绘制 Griewank 函数图形
x=[-8:0.1:8];
y=x;
[X,Y]=meshgrid(x,y);
[row,col]=size(X);
for l=1:col
   for h=1:row
      z(h,l)=Griewank([X(h,l),Y(h,l)]);
   end
end
surf(X,Y,z);
shading interp
```

将以上代码保存为 DrawGriewank.m 文件，并运行上述代码，得到 Griewank 函数图像，如图 10-3 所示。

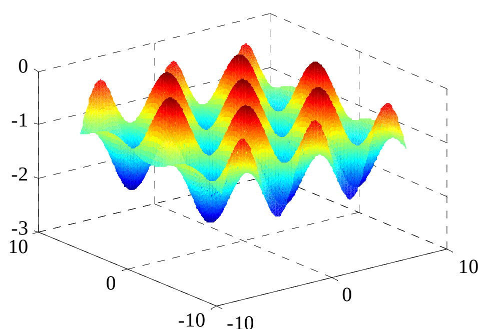


图 10-3 Griewank 函数图像

2.Rastrigin 函数

该函数的 MATLAB 代码如下所示：

```
function y=Rastrigin(x)
%Rastrigin 函数
%输入 x,给出相应的 y 值,在 x=(0,0,…,0)处有全局极小点 0
[row,col]=size(x);
if row>1
   error('输入的参数错误');
end
y=sum(x.^2-10*cos(2*pi*x)+10);
y=-y;
```

绘制以上函数图形的 MATLAB 代码：

```
function DrawRastrigin()
x=[-4:0.05:4];
y=x;
```

```
[X,Y]=meshgrid(x,y);
[row,col]=size(X);
for l=1:col
   for h=1:row
      z(h,l)=Rastrigin([X(h,l),Y(h,l)]);
   end
end
surf(X,Y,z);
shading interp
```

运行上述代码，得到 Rastrigin 函数图像，如图 10-4 所示。

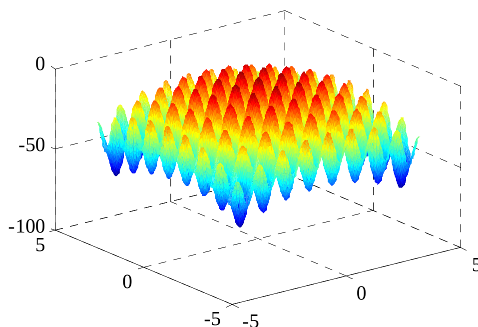


图 10-4　Rastrigin 函数图像

例 10-1，利用上文介绍的基本粒子群算法求解下列函数的最小值。

$$f(x)=\sum_{i=1}^{30} x_i^2+x_i-6$$

利用 PSO 算法求解最小值，首先需要确认不同迭代步数对结果的影响。设定题中函数的最小点均为 0，粒子群规模为 50，惯性权值为 0.5，学习因子 c_1 为 1.5，学习因子 c_2 为 2.5，迭代步数分别取为 100、1000、10000。

在 MATLAB 中建立目标函数代码：

```
function F=fitness(x)
F=0;
for i=1:30
   F=F+x(i)^2+x(i)-6
end
```

在 MATLAB 命令行窗口中输入代码：

```
>>x=zeros(1,30);
>> [xm1,fv1]=PSO(@fitness,50,1.5,2.5,0.5,100,30);
>> [xm2,fv2]=PSO(@fitness,50,1.5,2.5,0.5,1000,30);
>> [xm3,fv3]=PSO(@fitness,50,1.5,2.5,0.5,10000,30);
```

运行以上代码，比较目标函数取最小值时的自变量，如表 10-1 所示。

表 10-1　比较不同迭代步数下的目标函数值和最小值

迭代步数	100	1000	10000
x_1	-0.203470374827947	-0.322086628880754	-0.435569044575185
x_2	0.0614316795653017	-0.236499213027137	-0.456597706978179

（续表）

迭代步数	100	1000	10000
x_3	-0.432057786059138	-0.174672457595542	-0.272548635326235
x_4	-0.562337192754589	-0.323434573711674	-0.410028636513352
x_5	0.216285572045985	-0.559755785428548	-0.478395017745105
x_6	-0.448174496712675	-0.500724696101979	-0.438617720718304
x_7	0.0101008034620691	-0.334601378057723	-0.624351586431356
x_8	-0.359780035841033	-0.599261558115410	-0.542835397138839
x_9	-0.244678550463580	-0.689138008554286	-0.243113131019114
x_{10}	-0.316139905200595	-0.0694954358421096	-0.143940233374031
x_{11}	-0.408639179789461	-0.259841700046576	-0.706252186322252
x_{12}	-0.642619836410718	-0.246141170661282	-0.00781653355911016
x_{13}	-0.522925465434690	-0.449585957090094	-0.334838983888102
x_{14}	-0.203441587074036	-0.406235920268046	-0.104353647362726
x_{15}	-0.563308887343590	0.0891778287549033	-0.438931696076205
x_{16}	-0.301808274435673	-0.0303852886125965	-0.177440809228911
x_{17}	-0.709768167671245	-0.552156841443132	-0.621428723324555
x_{18}	-0.420233565717631	-0.354652539291389	-0.321409146325643
x_{19}	-0.0649786155553592	-0.473586592491481	-0.340215630334193
x_{20}	0.0835405545618331	-0.542947832512436	-0.435868961230739
x_{21}	-0.677113366792996	-0.571165888709759	-0.402359314141048
x_{22}	-0.288800585542166	-0.235227313009656	-0.663112621839921
x_{23}	-0.423115455971755	-0.783184021012424	-0.243847375888005
x_{24}	-0.483611573904200	-0.610977611626016	-0.372767988055409
x_{25}	-0.296101193584627	-0.0762667490397894	-0.588328193723098
x_{26}	-0.364523672340500	-0.389593896038030	-0.310699752647837
x_{27}	-0.217234643531979	-0.152204938081090	-0.474445660596261
x_{28}	0.0562371091188502	-0.812638082215613	-0.0836301944341218
x_{29}	-0.507805752603469	-0.661787823700067	-0.0284228008093966
x_{30}	-0.0208750670471909	-0.145197593442009	-0.397530666505423
目标函数最小值 fv	-184.692117109108	-184.692117109108	-184.692117109108

从表 10-1 可以看出，迭代步数不一定与获得解的精度成正比，即迭代步数越大，获得解的精度不一定越高。这是因为 PSO 算法是一种随机算法，同样的参数也会算出不同的结果。

在上述参数基础上，保持惯性权重 0.5、学习因子 c_1 为 1.5、学习因子 c_2 为 2.5、迭代步数为 100 不变，粒子群规模分别取 10、100 和 500，运行以下 MATLAB 代码：

```
>>x=zeros(1,30);
>> [xm1,fv1]=PSO(@fitness,10,1.5,2.5,0.5,100,30);
>> [xm2,fv2]=PSO(@fitness,100,1.5,2.5,0.5,100,30);
>> [xm3,fv3]=PSO(@fitness,500,1.5,2.5,0.5,100,30);
```

比较目标函数取最小值时的自变量，如表 10-2 所示。

表 10-2　比较不同粒子群规模下的目标函数值和最小值

粒子群规模	10	100	500
x_1	-0.461280538391346	-0.568652265006235	-0.490268156078420
x_2	-0.408370995746921	-0.452788770991822	-0.495317061863384
x_3	-0.0288416005345963	-0.388174768325847	-0.508017090877808
x_4	-0.0552338567227231	-0.401507545198533	-0.517007413849568
x_5	0.0738166644789645	-0.551259879300365	-0.477354073247202
x_6	-0.280868118500682	-0.233393064263199	-0.496014962954584
x_7	-0.429600925039530	-0.271896675443476	-0.489607302876620
x_8	-0.409562596099239	-0.547844449351226	-0.493034422510953
x_9	0.281766017074388	-0.380278337003657	-0.491570275741791
x_{10}	-0.587883598964542	-0.408568766862200	-0.505298045536549
x_{11}	-0.749463199823461	-0.626782730867803	-0.503117287033364
x_{12}	0.0779478416748528	-0.349408282182953	-0.494031258256908
x_{13}	-0.758300631146907	-0.583408316780879	-0.500060685658700
x_{14}	0.180131709578965	-0.375383139040645	-0.511709156436812
x_{15}	-0.564532674933458	-0.490162739466452	-0.517812810910794
x_{16}	-0.0637266236855537	-0.555105474483478	-0.504355035662881
x_{17}	0.501801473477060	-0.560793363305467	-0.511495990026503
x_{18}	0.583049171640106	-0.641197096800355	-0.519087838941761
x_{19}	0.423066993306820	-0.594790333100089	-0.497402575677108
x_{20}	0.463031353118403	-0.517368663564588	-0.506039272612501
x_{21}	-0.226652573205321	-0.647922715489912	-0.493311227454402
x_{22}	-0.340694973324611	-0.493043901761973	-0.492860555794895
x_{23}	0.303590596927068	-0.445059333754872	-0.499654192041048
x_{24}	-0.0372694887364219	-0.602557014069339	-0.494888427804042
x_{25}	-0.119240515687260	-0.439982689177553	-0.519431562496152
x_{26}	0.511293600728549	-0.260811072394469	-0.493925264779633
x_{27}	0.115534647931772	-0.738686510406502	-0.488810925337222
x_{28}	0.559536823964912	-0.494057140638969	-0.489181575636495
x_{29}	0.446461621552828	-0.378395529426522	-0.498224198470959
x_{30}	-0.359535394729040	-0.402673857684666	-0.514332244824747
目标函数最小值 fv	-176.440172293181	-187.045295621546	-187.496699775657

从表 10-2 中可以看出，粒子群规模越大，获得解的精度不一定越高。

综合以上不同迭代步数和不同粒子群规模运算得到的结果可知，在粒子群算法中，要想获得精度高的解，关键是各个参数之间的合适搭配。

10.1.3　经典应用

粒子群算法经常与其他算法混合使用。混合策略就是将其他进化算法、传统优化算法或其他技术应用到 PSO 中，用于提高粒子多样性、增强粒子的全局探索能力，或者提高局部开发能力、增强收敛速度与精度。

常用的粒子群混合方法基于免疫的粒子群算法。该算法是在免疫算法的基础上采用粒子群优化对抗体群体进行更新，可以解决免疫算法收敛速度慢的缺点。

基于免疫的混合粒子群算法步骤如下所示：

①确定学习因子 c_1 和 c_2、粒子（抗体）群体个数 M。

②由 logistic 回归分析映射产生 M 个粒子（抗体）x_i 及其速度 v_i，其中 i=1,…,N，最后形成初始粒子（抗体）群体 P_0。

③生产免疫记忆粒子（抗体）：计算当前粒子（抗体）群体 P 中粒子（抗体）的适应值并判断算法是否满足结束条件，如果满足则结束并输出结果，否则继续运行。

④更新局部和全局最优解，并根据下面的公式更新粒子位置和速度。

$$x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1)\ ,\ \ j=1 \text{L}\ d$$

$$v_{i,j}(t+1)=w\times v_{i,j}(t)+c_1r_1[p_{i,j}-x_{i,j}(t)]+c_2r_2[p_{g,j}-x_{i,j}(t)]$$

⑤由 logistic 映射产生 N 个新的粒子（抗体）。

⑥基于浓度的粒子（抗体）选择：用群体中相似抗体百分比计算生产 $N+M$ 个新粒子（抗体）的概率，依照概率大小选择 N 个粒子（抗体）形成粒子（抗体）群 P。然后转入第③步。

算法流程图如图 10-5 所示。

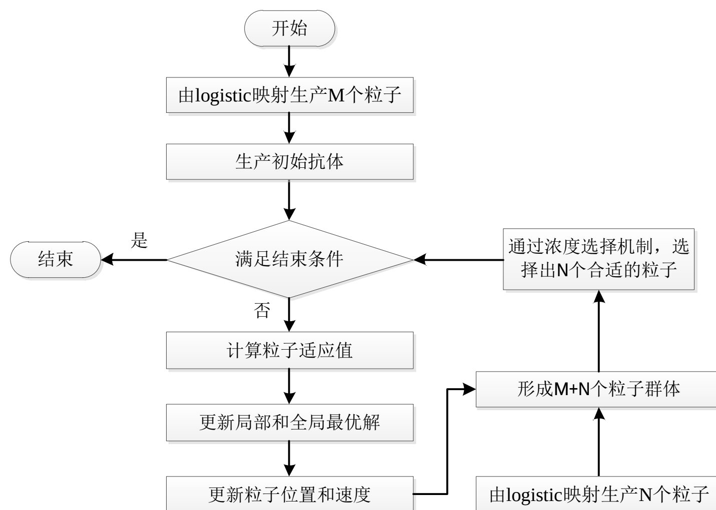


图 10-5　免疫粒子群算法流程

将实现自适应权重的优化函数命名为 PSO_immu，在 MATLAB 中编写实现以上步骤的代码：

```
function [x,y,Result]=PSO_immu(func,N,c1,c2,w,MaxDT,D,eps,DS,replaceP,minD,Psum)
format long;
%%%%%%给定初始化条件%%%%%%%%%%%%%%%%%%%%%%%%%%%
% c1=2;                     %学习因子 1
% c2=2;                     %学习因子 2
% w=0.8;                    %惯性权重
% MaxDT=100;                %最大迭代次数
% D=2;                      %搜索空间维数（未知数个数）
% N=100;                    %初始化群体个体数目
% eps=10^(-10);             %设置精度(在已知最小值时用)
% DS=8;                     %每隔 DS 次循环就检查最优个体是否变优
% replaceP=0.5;             %粒子的概率大于 replaceP 将被免疫替换
% minD=1e-10;               %粒子间的最小距离
% Psum=0;                   %个体最佳的和
range=100;
count = 0;
%%%%%%初始化种群的个体%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:N
   for j=1:D
      x(i,j)=-range+2*range*rand;     %随机初始化位置
      v(i,j)=randn;                   %随机初始化速度
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%先计算各个粒子的适应度，并初始化 Pi 和 Pg%%%%%%%%%%%%%%%%%%%%%%
for i=1:N
   p(i)=feval(func,x(i,:));

   y(i,:)=x(i,:);
end
pg=x(1,:);                  %Pg 为全局最优
for i=2:N
   if feval(func,x(i,:))<feval(func,pg)
      pg=x(i,:);
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%主循环，按照公式依次迭代，直到满足精度要求%%%%%%%%
for t=1:MaxDT
   for i=1:N
      v(i,:)=w*v(i,:)+c1*rand*(y(i,:)-x(i,:))+c2*rand*(pg-x(i,:));
      x(i,:)=x(i,:)+v(i,:);
      if feval(func,x(i,:))<p(i)
         p(i)=feval(func,x(i,:));
         y(i,:)=x(i,:);
      end
      if p(i)<feval(func,pg)
         pg=y(i,:);
         subplot(1,2,1);
         bar(pg,0.25);
         axis([0 3 -40 40 ]) ;
         title (['Iteration  ', num2str(t)]); pause (0.1);
         subplot(1,2,2);
        plot(pg(1,1),pg(1,2),'rs','MarkerFaceColor','r', 'MarkerSize',8)
         hold on;
         plot(x(:,1),x(:,2),'k.');
         set(gca,'Color','g')
```

```
          hold off;
          grid on;
          axis([-100 100 -100 100 ]) ;
          title(['Global Min =  ',num2str(p(i))]);
          xlabel(['Min_x= ',num2str(pg(1,1)),'  Min_y= ',num2str(pg(1,2))]);

        end
    end
    Pbest(t)=feval(func,pg) ;
%    if Foxhole(pg,D)<eps     %如果结果满足精度要求则跳出循环
%       break;
%    end
%%%%%开始进行免疫%%%%%%%%%%%%%%%
    if t>DS
       if mod(t,DS)==0 && (Pbest(t-DS+1)-Pbest(t))<1e-020
          %如果连续 DS 代数，群体中的最优没有明显变化，则进行免疫
%在函数测试的过程中发现，经过一定代数的更新，个体最优不完全相等，但变化非常非常小
          for i=1:N             %先计算出个体最优的和
            Psum=Psum+p(i);
          end

          for i=1:N             %免疫程序

             for j=1:N          %计算每个个体与个体 i 的距离
                distance(j)=abs(p(j)-p(i));
             end
             num=0;
             for j=1:N          %计算与第 i 个个体距离小于 minD 的个数
                if distance(j)<minD
                   num=num+1;
                end
             end
             PF(i)=p(N-i+1)/Psum;   %计算适应度概率
             PD(i)=num/N;       %计算个体浓度

             a=rand;            %随机生成计算替换概率的因子
             PR(i)=a*PF(i)+(1-a)*PD(i);      %计算替换概率
          end

           for i=1:N
              if PR(i)>replaceP
                 x(i,:)=-range+2*range*rand(1,D);
             count=count+1;
              end
          end
       end
    end
end

%%%%%%%最后给出计算结果%%%%%%%%%%%%%%%%%%%%
x=pg(1,1);
y=pg(1,2);
Result=feval(func,pg);
%%%%%%%%%%算法结束%%%%%%%%%%%%%%%%%%
function probabolity(N,i)
PF=p(N-i)/Psum;%适应度概率
disp(PF);
```

```
for jj=1:N
  distance(jj)=abs(P(jj)-P(i));
end
num=0;
for ii=1:N
    if distance(ii)<minD
        num=num+1;
    end
end
PD=num/N;                    %个体浓度
PR=a*PF+(1-a)*PD;       %替换概率
```

例 10-2，使用基于模拟退火的混合粒子群算法，求解函数 $f(x)=\frac{\cos\sqrt{x_1^2+x_2^2}-1}{[1+(x_1^2-x_2^2)]^2}+0.5$ 最小值。其中 $-10\leqslant x_i\leqslant 10$，粒子数为 50，学习因子均为 2，退火常数取值 0.6，迭代步数为 1000。

首先建立目标函数代码：

```
function y=immuFunc(x)
   y=(cos(x(1)^2+ x(2)^2)-1)/((1+ (x(1)^2- x(2)^2))^2)+0.5;
end
```

在 MATLAB 命令行窗口中输入代码：

```
[xm,fv] = PSO_immu (@immuFunc,50,2,2,0.8,100,5,0.0000001,10,0.6,
    0.0000000000000000001,0)
```

运行结果：

```
>>xm =
   1.139888959718036
fv =
  -1.515992561220435
```

得到目标函数取最小值时的自变量 xm 变化图，如图 10-6 所示。

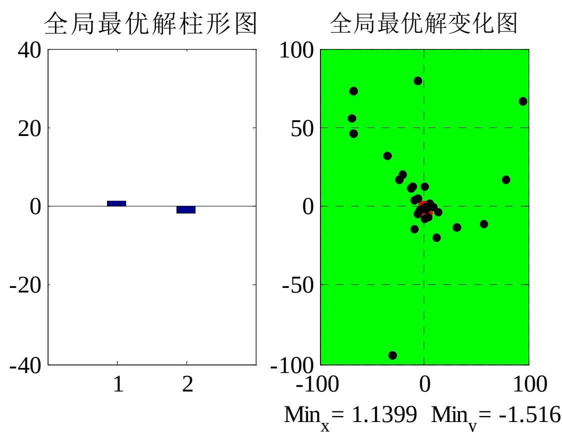


图 10-6　目标函数取最小值时的自变量 xm 变化图

10.2　遗传算法的 MATLAB 实现

遗传算法（Genetic Algorithm）是模拟自然界生物进化机制的一种算法，即遵循适者生存、优胜劣汰的法则，也就是寻优过程中有用的保留无用的则去除。在科学和生产实践中表现为在所有可能的解决方法中找出最符合该问题所要求的条件的解决方法，即找出一个最优解。

10.2.1　基本原理

遗传操作是模拟生物基因遗传的做法。在遗传算法中，通过编码组成初始群体后，遗传操作的任务就是对群体的个体按照它们对环境适应度(适应度评估)施加一定的操作，从而实现优胜劣汰的进化过程。从优化搜索的角度而言，遗传操作可使问题的解，一代又一代地优化，并逼近最优解。遗传算法过程如图 10-7 所示。

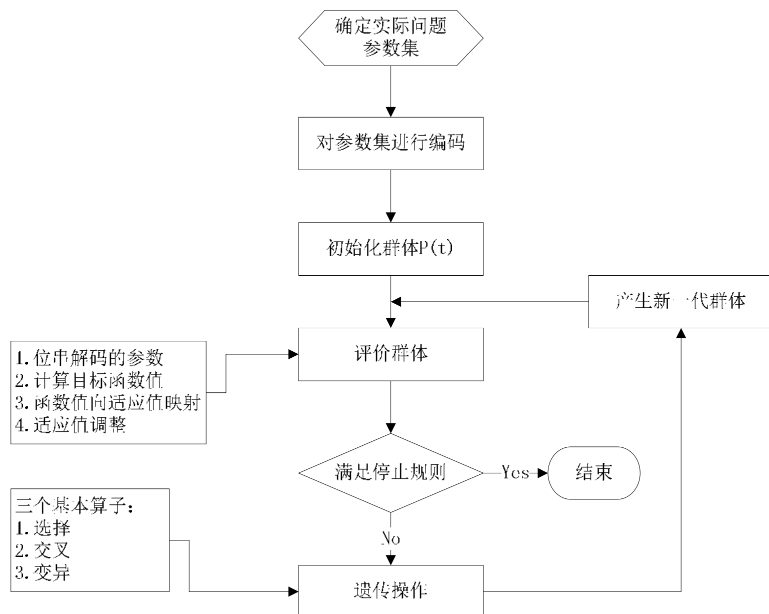


图 10-7　遗传算法过程

遗传操作包括以下三个基本遗传算子（genetic operator）：选择（selection）、交叉（crossover）和变异（mutation）。

个体遗传算子的操作都是在随机扰动情况下进行的。因此，群体中个体向最优解迁移的规则是随机的。需要强调的是，这种随机化操作和传统的随机搜索方法是有区别的。遗传操作进行高效有向的搜索，而不是如一般随机搜索方法进行无向搜索。

遗传操作的效果和上述三个遗传算子所取的操作概率、编码方法、群体大小、初始群体，以及适应度函数的设定密切相关。

1. 选择

从群体中选择优胜的个体，淘汰劣质个体的操作叫选择。选择算子有时又称为再生算子（reproduction operator）。选择的目的是把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。

轮盘赌选择法（roulette wheel selection）是最简单也是最常用的选择方法。在该方法中，各个个体的选择概率和其适应度值成比例。设群体大小为 n，其中个体 i 的适应度为 f_i，则 i 被选择的概率为 $P_i = f_i / \sum_{j=1}^{n} f_i$。

显然，概率反映了个体 i 的适应度在整个群体的个体适应度总和中所占的比例。个体适应度越大，其被选择的概率就越大，反之亦然。

计算出群体中各个个体的选择概率后，为了选择交配个体，需要进行多轮选择。每一轮产生一个[0,1]之间的均匀随机数，将该随机数作为选择指针来确定被选个体。

个体被选后，可随机地组成交配对，以供后面的交叉操作。

2. 交叉

在自然界生物进化过程中起核心作用的是生物遗传基因的重组(加上变异)。同样，遗传算法中起核心作用的是遗传操作的交叉算子。所谓交叉是指把两个父代个体的部分结构加以替换重组而生成新个体的操作。通过交叉，遗传算法的搜索能力得以飞跃提高。

交叉算子根据交叉率将种群中的两个个体随机地交换某些基因，能够产生新的基因组合，期望将有益基因组合在一起。根据编码表示方法的不同，可以有以下算法：

①实值重组（real valued recombination）

- 离散重组（discrete recombination）
- 中间重组（intermediate recombination）
- 线性重组（linear recombination）
- 扩展线性重组（extended linear recombination）

②二进制交叉（binary valued crossover）

- 单点交叉（single-point crossover）
- 多点交叉（multiple-point crossover）
- 均匀交叉（uniform crossover）
- 洗牌交叉（shuffle crossover）
- 缩小代理交叉（crossover with reduced surrogate）

最常用的交叉算子为单点交叉(one-point crossover)。具体操作是:在个体串中随机设定一个交叉点，实行交叉时，该点前或后的两个个体的部分结构进行互换，并生成两个新个体。下面给出单点交叉的一个例子：

个体 A：1 0 0 1 ↑1 1 1 → 1 0 0 1 0 0 0 新个体

个体 B：0 0 1 1 ↑0 0 0 → 0 0 1 1 1 1 1 新个体

3. 变异

变异算子的基本内容是对群体中的个体串的某些基因座上的基因值做变动。依据个体编码表示方法的不同，可以有以下算法：

- 实值变异
- 二进制变异

一般来说，变异算子操作的基本步骤如下：

- 对群中所有个体以事先设定的变异概率判断是否进行变异
- 对进行变异的个体随机选择变异位进行变异

遗传算法引入变异的目的有两个：一是使遗传算法具有局部的随机搜索能力。当遗传算法通过交叉算子已接近最优解邻域时，利用变异算子的这种局部随机搜索能力可以加速向最优解收敛。

显然，此种情况下的变异概率应取较小值，否则接近最优解的积木块会因变异而遭到破坏。二是使遗传算法可维持群体多样性，以防止出现未成熟收敛现象。此时收敛概率应取较大值。

遗传算法中，交叉算子因其全局搜索能力而作为主要算子，变异算子因其局部搜索能力而作为辅助算子。

遗传算法通过交叉和变异这对相互配合又相互竞争的操作而使其具备兼顾全局和局部的均衡搜索能力。

所谓相互配合，是指当群体在进化中陷于搜索空间中某个超平面而仅靠交叉不能摆脱时，通过变异操作可有助于这种摆脱。

所谓相互竞争，是指当通过交叉已形成所期望的积木块时，变异操作有可能破坏这些积木块。如何有效地配合使用交叉和变异操作，是目前遗传算法的一个重要研究内容。

基本变异算子是指对群体中的个体码串随机挑选一个或多个基因座并对这些基因座的基因值做变动，(0，1)二值码串中的基本变异操作如下：

$$(\text{个体}A)\ 10\underset{*}{0}1\underset{*}{0}110 \xrightarrow{\text{变异}} 11\underset{*}{0}0\underset{*}{0}110\ (\text{个体}A')$$

> **注意**
>
> 在基因位下方标有*号的基因发生变异。

变异率的选取一般受种群大小、染色体长度等因素的影响，通常选取很小的值，一般取0.001～0.1。

4. 终止条件

当最优个体的适应度达到给定的阈值，或者最优个体的适应度和群体适应度不再上升时，或者迭代次数达到预设的代数时，算法终止。预设的代数一般设置为100～500代。

10.2.2　程序设计

为了更好地在 MATLAB 中使用遗传算法，本节主要对遗传算法的程序设计和 MATLAB 工具箱做讲解。

随机初始化种群 $P(t)=\{x_1,x_2,\cdots,x_n\}$，计算 P(t)中个体的适应值。其 MATLAB 程序的基本格式如下所示：

```
Begin
t=0
初始化 P(t)
计算 P(t)的适应值;
while (不满足停止准则)
    do
    begin
    t=t+1
    从 P(t+1)中选择 P(t)
    重组 P(t)
计算 P(t)的适应值
end
```

例 10-3，求下列函数的最大值。

$f(x)=9*\sin(5x)+8*\cos(4x)$，其中 $x\in[0,15]$

1. 初始化

遗传算法 MATLAB 子程序如下：

```
%初始化
function pop=initpop(popsize,chromlength)
pop=round(rand(popsize,chromlength));
% rand 随机产生每个单元为{0,1}行数为 popsize，列数为 chromlength 的矩阵，
% roud 对矩阵的每个单元进行圆整。这样产生的初始种群。
end
```

- initpop.m 函数的功能是实现群体的初始化。
- popsize 表示群体的大小。
- chromlength 表示染色体的长度（二值数的长度），长度大小取决于变量的二进制编码的长度。

2. 目标函数值

①二进制数转化为十进制数

遗传算法 MATLAB 子程序如下：

```
function pop2=decodebinary(pop)
[px,py]=size(pop);
%求 pop 行和列数
for i=1:py
pop1(:,i)=2.^(py-i).*pop(:,i);
end
pop2=sum(pop1,2);
%求 pop1 的每行之和
end
```

②二进制编码转化为十进制数

decodechrom.m 函数的功能是将染色体（或二进制编码）转换为十进制，参数 spoint 表示待解码的二进制串的起始位置。

对于多个变量而言，如有两个变量，采用 20 位表示，每个变量 10 位，则第一个变量从 1 开始，另一个变量从 11 开始。参数 length 表示所截取的长度。

遗传算法 MATLAB 子程序如下：

```
%将二进制编码转换成十进制
function pop2=decodechrom(pop,spoint,length)
pop1=pop(:,spoint:spoint+length-1);
pop2=decodebinary(pop1);
end
```

③计算目标函数值

calobjvalue.m 函数的功能是实现目标函数的计算。

遗传算法 MATLAB 子程序如下：

```
function [objvalue]=calobjvalue(pop)
temp1=decodechrom(pop,1,10);        %将 pop 每行转化成十进制数
x=temp1*10/1023;                    %将二值域 中的数转化为变量域 的数
objvalue=10*sin(5*x)+7*cos(4*x);    %计算目标函数值
end
```

3. 计算个体的适应值

遗传算法 MATLAB 子程序如下：

```
%计算个体的适应值
function fitvalue=calfitvalue(objvalue)
global Cmin;
Cmin=0;
[px,py]=size(objvalue);
for i=1:px
    if objvalue(i)+Cmin>0
        temp=Cmin+objvalue(i);
    else
        temp=0.0;
    end
    fitvalue(i)=temp;
end
fitvalue=fitvalue'
```

4. 选择复制

选择复制操作是决定哪些个体可以进入下一代。程序中采用赌轮盘选择法选择，这种方法较易实现。根据方程 $p_i = f_i / \sum f_i = f_i / f_{sum}$，选择步骤：

①在第 t 代，计算 f_{sum} 和 p_i。

②产生{0,1}的随机数 rand(.)，求 s=rand(.)*f_{sum}。

③求 $\sum_{i=1}^{k} f_i \geq s$ 中最小的 k，则第 k 个个体被选中。

④进行 N 次②、③操作，得到 N 个个体，成为第 $t=t+1$ 代种群。

遗传算法 MATLAB 子程序如下：

```
%选择复制
function [newpop]=selection(pop,fitvalue)
totalfit=sum(fitvalue);       %求适应值之和
fitvalue=fitvalue/totalfit;   %单个个体被选择的概率
fitvalue=cumsum(fitvalue);    %如 fitvalue=[1 2 3 4]，则 cumsum(fitvalue)=[1 3 6 10]
[px,py]=size(pop);
ms=sort(rand(px,1));          %从小到大排列
fitin=1;
newin=1;
while newin<=px
   if(ms(newin))<fitvalue(fitin)
      newpop(newin)=pop(fitin);
      newin=newin+1;
   else
      fitin=fitin+1;
   end
end
```

5. 交叉

交叉（crossover），群体中的每个个体之间都以一定的概率 pc 交叉，即两个个体从各自字符串的某一位置（一般是随机确定）开始互相交换，这类似于生物进化过程中的基因分裂与重组。

例如，假设 2 个父代个体 x1，x2 为：

x1=0100110

x2=1010001

从每个个体的第 3 位开始交叉，交叉后得到 2 个新的子代个体 y1，y2：

y1＝0100001

y2＝1010110

这样 2 个子代个体就分别具有了 2 个父代个体的某些特征。

利用交叉我们有可能由父代个体在子代组合成具有更高适合度的个体。事实上交叉是遗传算法区别于其他传统优化方法的主要特点之一。

遗传算法 MATLAB 子程序如下：

```
%交叉
function [newpop]=crossover(pop,pc)
[px,py]=size(pop);
newpop=ones(size(pop));
for i=1:2:px-1
   if(rand<pc)
      cpoint=round(rand*py);
      newpop(i,:)=[pop(i,1:cpoint),pop(i+1,cpoint+1:py)];
      newpop(i+1,:)=[pop(i+1,1:cpoint),pop(i,cpoint+1:py)];
   else
      newpop(i,:)=pop(i);
      newpop(i+1,:)=pop(i+1);
   end
end
```

6. 变异

变异（mutation），基因的突变普遍存在于生物的进化过程中。变异是指父代中的每个个体的每一位都以概率 pm 翻转，即由“1”变为“0”，或由“0”变为“1”。

遗传算法的变异特性可以使求解过程随机地搜索到解可能存在的整个空间，因此可以在一定程度上求得全局最优解。

遗传算法 MATLAB 子程序如下：

```
%变异
function [newpop]=mutation(pop,pm)
[px,py]=size(pop);
newpop=ones(size(pop));
for i=1:px
   if(rand<pm)
      mpoint=round(rand*py);
      if mpoint<=0
         mpoint=1;
      end
      newpop(i)=pop(i);
      if any(newpop(i,mpoint))==0
         newpop(i,mpoint)=1;
      else
         newpop(i,mpoint)=0;
      end
   else
      newpop(i)=pop(i);
   end
end
```

7. 求出群体中最大的适应值及其个体

遗传算法 MATLAB 子程序如下：

```
%求出群体中适应值最大的值
function [bestindividual,bestfit]=best(pop,fitvalue)
[px,py]=size(pop);
bestindividual=pop(1,:);
bestfit=fitvalue(1);
for i=2:px
   if fitvalue(i)>bestfit
      bestindividual=pop(i,:);
      bestfit=fitvalue(i);
   end
end
```

8. 主程序

遗传算法 MATLAB 主程序如下：

```
clear all
clc
popsize=20;                                 %群体大小
chromlength=10;                             %字符串长度（个体长度）
pc=0.7;                                     %交叉概率
pm=0.005;                                   %变异概率
pop=initpop(popsize,chromlength);           %随机产生初始群体
for i=1:20                                  %20 为迭代次数
[objvalue]=calobjvalue(pop);                %计算目标函数
```

```
    fitvalue=calfitvalue(objvalue);           %计算群体中每个个体的适应度
    [newpop]=selection(pop,fitvalue);         %复制
    [newpop]=crossover(pop,pc);               %交叉
    [newpop]=mutation(pop,pc);                %变异
    [bestindividual,bestfit]=best(pop,fitvalue);    %求出群体中适应值最大的个体及其适应值
    y(i)=max(bestfit);
    n(i)=i;
    pop5=bestindividual;
    x(i)=decodechrom(pop5,1,chromlength)*10/1023;
    pop=newpop;
    end
    fplot('9*sin(5*x)+8*cos(4*x)',[0 15])
    hold on
    plot(x,y,'r*')
    hold off
```

运行主程序，得到结果如图 10-8 所示。

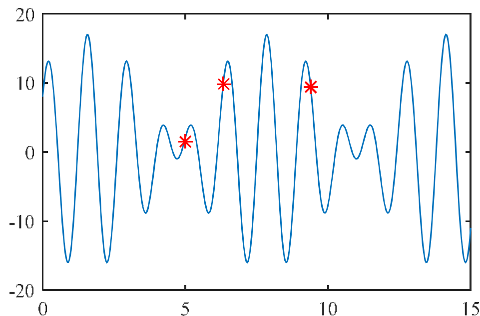


图 10-8　遗传算法仿真结果

注意

遗传算法中有 4 个参数需要提前设定，一般在以下范围内进行设置：

- 群体大小：20~100
- 遗传算法的终止进化代数：100~500
- 交叉概率：0.4~0.99
- 变异概率：0.0001~0.1

10.2.3　经典应用

旅行商问题（Traveling Salesman Problem，TSP），也称货郎担问题，是数学领域中的著名问题之一。TSP 问题已经被证明是一个 NP-hard 问题，由于 TSP 问题代表一类组合优化问题，因此对其近似解的研究一直是算法设计的一个重要问题。

TSP 问题从描述上来看是一个非常简单的问题，给定 n 个城市和各城市之间的距离，寻找一条遍历所有城市且每个城市只被访问一次的路径。并保证总路径距离最短。其数学描述如下：

设 $G = (V,E)$为赋权图，V ={1,2,⋯,n}为顶点集，E 为边集，各顶点间距离为 C_{ij}，已知 $C_{ij} > 0$，且 $i, j \in V$，并设定：

$$x_{ij}=\begin{cases}1 & \text{最优路径}\\ 0 & \text{其他情况}\end{cases}$$

那么整个 TSP 问题的数学模型表示如下：

$$\min Z=\sum_{i\neq j} C_{ij}x_{ij}$$

$$\begin{cases}\sum_{i\neq j} x_{ij}=1, & j\in v\\ \sum_{i,j\in s} x_{ij}\leq |k|-1, & k\subset v\end{cases}, \text{其中} x_{ij}\in\{0,1\}, i\in v, j\in v$$

其中，k 是 v 的全部非空子集，$|k|$是集合 k 中包含图 G 的全部顶点的个数。

遗传算法求解 TSP 的基本步骤：

（1）种群初始化：个体编码方法有二进制编码和实数编码，在解决 TSP 问题过程中个体编码方法为实数编码。对于 TSP 问题，实数编码为 1-n 的实数的随机排列，初始化的参数有种群个数 M、染色体基因个数 N（即城市的个数）、迭代次数 C、交叉概率 Pc、变异概率 Pmutation。

（2）适应度函数：在 TSP 问题中，每个染色体（即 n 个城市的随机排列）可计算出总距离，因此可将一个随机全排列的总距离的倒数作为适应度函数，即距离越短，适应度函数越好，满足 TSP 要求。

（3）选择操作：遗传算法选择操作有轮盘赌法、锦标赛法等多种方法，用户可根据实际情况选择最合适的算法。

（4）交叉操作：遗传算法中交叉操作有多种方法。一般对于个体，可以随机选择两个个体，在对应位置交换若干个基因片段，同时保证每个个体依然是 1-n 的随机排列，防止进入局部收敛。

（5）变异操作：对于变异操作，随机选取个体，同时随机选取个体的两个基因进行交换以实现变异操作。

例 10-4，随机生成一组城市种群，利用遗传算法寻找一条遍历所有城市且每个城市只被访问一次的路径，且总路径距离最短的方法。

根据分析，完成 MATLAB 主函数：

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%主函数%%%%%%%%%%%%%%%%%%%%
clear;
clc;
%%%%%%%%%%%%%%输入参数%%%%%%%%%%%%%%%%%%
N=10;                    %%城市的个数
M=20;                    %%种群的个数
C=100;                   %%迭代次数
C_old=C;
m=2;                     %%适应值归一化淘汰加速指数
Pc=0.4;                  %%交叉概率
Pmutation=0.2;           %%变异概率

%%%%%%%%%%%生成城市的坐标%%%%%%%%%%%%%%%%%%%%%%%%
pos=randn(N,2);
```

```
%%%%%%%%生成城市之间距离矩阵%%%%%%%%%%%%
D=zeros(N,N);
for i=1:N
   for j=i+1:N
      dis=(pos(i,1)-pos(j,1)).^2+(pos(i,2)-pos(j,2)).^2;
      D(i,j)=dis^(0.5);
      D(j,i)=D(i,j);
   end
end

%%%%%%%%生成初始群体%%%%%%%%%%%%%%%%%%%%
popm=zeros(M,N);
for i=1:M
   popm(i,:)=randperm(N);
end

%%%%%%%%随机选择一个种群%%%%%%%%%%%%%%%%%
R=popm(1,:);
figure(1);
scatter(pos(:,1),pos(:,2) ,'k.');
xlabel('横轴')
ylabel('纵轴')
title('随机产生的种群图')
axis([-3 3 -3 3]);
figure(2);
plot_route(pos,R);
xlabel('横轴')
ylabel('纵轴')
title('随机生成种群中城市路径情况')
axis([-3 3 -3 3]);

%%%%%%%%初始化种群及其适应函数%%%%%%%%%%%%
fitness=zeros(M,1);
len=zeros(M,1);
for i=1:M
   len(i,1)=myLength(D,popm(i,:));
end
maxlen=max(len);
minlen=min(len);
fitness=fit(len,m,maxlen,minlen);
rr=find(len==minlen);
R=popm(rr(1,1),:);
for i=1:N
fprintf('%d ',R(i));
end
fprintf('\n');
fitness=fitness/sum(fitness);
distance_min=zeros(C+1,1);        %%各次迭代的最小的种群的距离
while C>=0
fprintf('迭代第%d 次\n',C);
%%%%选择操作%%%%
nn=0;
for i=1:size(popm,1)
   len_1(i,1)=myLength(D,popm(i,:));
   jc=rand*0.3;
   for j=1:size(popm,1)
```

```
            if fitness(j,1)>=jc
            nn=nn+1;
            popm_sel(nn,:)=popm(j,:);
            break;
            end
        end
    end
    %%%%每次选择都保存最优的种群%%%%
    popm_sel=popm_sel(1:nn,:);
    [len_m len_index]=min(len_1);
    popm_sel=[popm_sel;popm(len_index,:)];
    %%%%交叉操作%%%%
    nnper=randperm(nn);
    A=popm_sel(nnper(1),:);
    B=popm_sel(nnper(2),:);
    for i=1:nn*Pc
    [A,B]=cross(A,B);
    popm_sel(nnper(1),:)=A;
    popm_sel(nnper(2),:)=B;
    end
    %%%%变异操作%%%%
    for i=1:nn
        pick=rand;
        while pick==0
            pick=rand;
        end
        if pick<=Pmutation
          popm_sel(i,:)=Mutation(popm_sel(i,:));
        end
    end
    %%%%求适应度函数%%%%
    NN=size(popm_sel,1);
    len=zeros(NN,1);
    for i=1:NN
        len(i,1)=myLength(D,popm_sel(i,:));
    end
    maxlen=max(len);
    minlen=min(len);
    distance_min(C+1,1)=minlen;
    fitness=fit(len,m,maxlen,minlen);
    rr=find(len==minlen);
    fprintf('minlen=%d\n',minlen);
    R=popm_sel(rr(1,1),:);
    for i=1:N
    fprintf('%d ',R(i));
    end
    fprintf('\n');
    popm=[];
    popm=popm_sel;
    C=C-1;
    %pause(1);
    end
    figure(3)
    plot_route(pos,R);
    xlabel('横轴')
    ylabel('纵轴')
    title('优化后的种群中城市路径情况')
```

```
axis([-3 3 -3 3]);
```

主函数中用到的函数代码：

（1）适应度函数代码。

```
%%%%%%%%适应度函数%%%%%%%%%%%%%%%%%%%%%%
function fitness=fit(len,m,maxlen,minlen)
   fitness=len;
   for i=1:length(len)
      fitness(i,1)=(1-(len(i,1)-minlen)/(maxlen-minlen+0.0001)).^m;
   end
end
```

（2）计算个体距离函数代码。

```
%%%%%%%%计算个体距离函数%%%%%%%%%%%%%
function len=myLength(D,p)
   [N,NN]=size(D);
   len=D(p(1,N),p(1,1));
   for i=1:(N-1)
      len=len+D(p(1,i),p(1,i+1));
   end
end
```

（3）交叉操作函数代码。

```
%%%%%%%%交叉操作函数%%%%%%%%%%%%%%%%%%%%%%
function [A,B]=cross(A,B)
   L=length(A);
   if L<10
      W=L;
   elseif ((L/10)-floor(L/10))>=rand&&L>10
      W=ceil(L/10)+8;
   else
      W=floor(L/10)+8;
   end
   p=unidrnd(L-W+1);
   fprintf('p=%d ',p);
   for i=1:W
      x=find(A==B(1,p+i-1));
      y=find(B==A(1,p+i-1));
      [A(1,p+i-1),B(1,p+i-1)]=exchange(A(1,p+i-1),B(1,p+i-1));
      [A(1,x),B(1,y)]=exchange(A(1,x),B(1,y));
   end
end
```

（4）对调函数代码。

```
%%%%%%%%对调函数%%%%%%%%%%%%%%%%%%%%%%
function [x,y]=exchange(x,y)
   temp=x;
   x=y;
      y=temp;
end
```

（5）变异函数代码。

```
%%%%%%%%变异函数%%%%%%%%%%%%%%%%%%%%%%
function a=Mutation(A)
   index1=0;index2=0;
   nnper=randperm(size(A,2));
```

```
    index1=nnper(1);
    index2=nnper(2);
    %fprintf('index1=%d ',index1);
    %fprintf('index2=%d ',index2);
    temp=0;
    temp=A(index1);
    A(index1)=A(index2);
    A(index2)=temp;
    a=A;
end
```

（6）绘制连点画图函数代码。

```
%%%%%%%连点画图函数%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plot_route(a,R)
    scatter(a(:,1),a(:,2),'rx');
    hold on;
    plot([a(R(1),1),a(R(length(R)),1)],[a(R(1),2),a(R(length(R)),2)]);
    hold on;
    for i=2:length(R)
        x0=a(R(i-1),1);
        y0=a(R(i-1),2);
        x1=a(R(i),1);
        y1=a(R(i),2);
        xx=[x0,x1];
        yy=[y0,y1];
        plot(xx,yy);
        hold on;
    end
end
```

运行主程序，得到随机产生的城市种群图，如图 10-9 所示，随机生成种群中城市路径情况如图 10-10 所示。

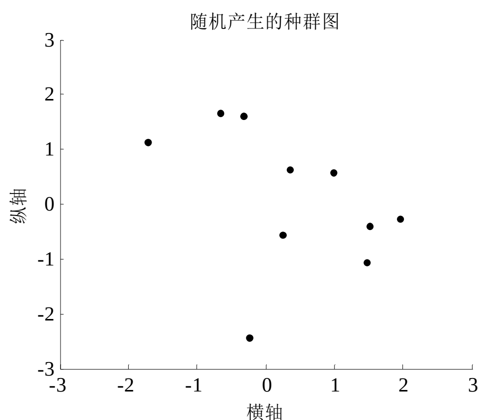


图 10-9　随机产生的城市种群图

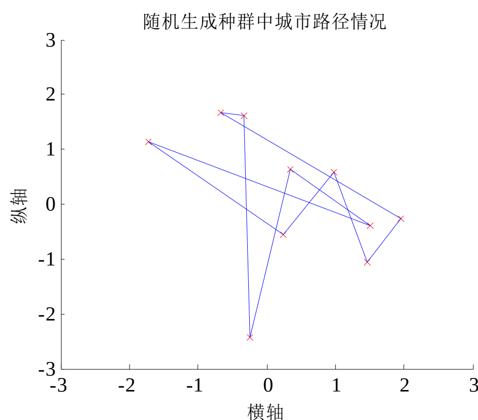


图 10-10　随机生成种群中城市路径情况

从图 10-9 中可以看出，随机产生的种群城市点不对称，也没有规律，用一般的方法很难得到其最优路径。从图 10-10 中可以看出，随机产生的路径长度很长，空行浪费比较多。

运行遗传算法，得到如图 10-11 所示的城市路径。从图中可以看出，该路径明显优于图 10-10 中的路径，且每个城市只经过一次。

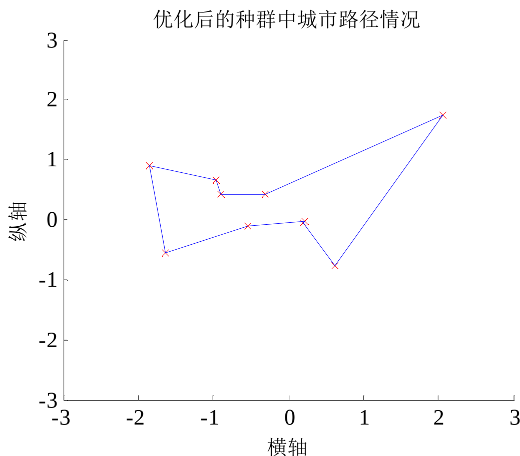


图 10-11　优化后的城市路径

10.3　蚁群算法概述

蚁群算法由 Dorigo 于 1992 年在他的博士论文中提出，其灵感来源于蚂蚁在寻找食物过程中发现路径的行为。

10.3.1　基本原理

蚁群优化算法是模拟蚂蚁觅食的原理，设计出的一种群集智能算法。蚂蚁在觅食过程中能够在其经过的路径上留下一种称为信息素的物质，并在觅食过程中能够感知这种物质的强度，并指导自己的行动方向，它们总是朝着该物质强度高的方向移动，因此大量蚂蚁组成的集体觅食就表现为一种对信息素的正反馈现象。

某一条路径越短，路径上经过的蚂蚁越多，其信息素遗留的也就越多，信息素的浓度也就越高，蚂蚁选择这条路径的概率也就越高，由此构成的正反馈过程，从而逐渐逼近最优路径，找到最优路径。

蚂蚁觅食的运行轨迹模式如图 10-12 所示。蚂蚁以信息素作为媒介而间接进行信息交流，判断洞穴到食物地点的最佳路径。

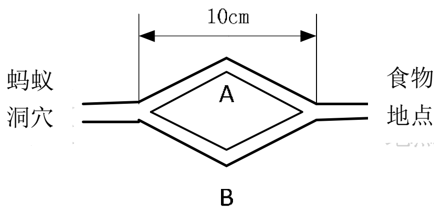


图 10-12　蚂蚁运行轨迹模式

当蚂蚁从食物源走到蚁穴，或者从蚁穴走到食物源时，都会在经过的路径上释放信息素，

从而形成一条含有信息素的路径，蚂蚁可以感觉出路径上信息素浓度的大小，并且以较高的概率选择信息素浓度较高的路径。

人工蚂蚁的搜索主要包括三种智能行为：

- 蚂蚁利用信息素进行相互通信：蚂蚁在所选择的路径上会释放一种叫作信息素的物质，当其他蚂蚁进行路径选择时，会根据路径上的信息素浓度进行选择，这样信息素就成为蚂蚁之间进行通信的媒介。
- 蚂蚁的记忆行为：一只蚂蚁搜索过的路径在下次搜索时就不再被该蚂蚁选择，因此在蚁群算法中建立禁忌表进行模拟。
- 蚂蚁的集群活动：通过一只蚂蚁的运动很难达到事物源，但整个蚁群进行搜索就完全不同。当某些路径上通过的蚂蚁越来越多时，路径上留下的信息素数量也就越多，导致信息素强度增大，蚂蚁选择该路径的概率随之增加，从而进一步增加该路径的信息素强度，而通过的蚂蚁比较少的路径上的信息素会随着时间的推移而挥发，从而变得越来越少。

10.3.2 程序设计

蚁群算法（ACO）不仅利用了正反馈原理，在一定程度上可以加快进化过程，而且是一种本质并行的算法，个体之间不断进行信息交流和传递，有利于发现较好解。

根据上一节对蚁群算法的介绍，编写蚁群算法 MATLAB 源程序：

```
clear all
clc
%%初始化%%%%%%%%%%%%%%%%%%%%%%%%%%
Ant=300;     %蚂蚁数量
Times=80;    %蚂蚁移动次数
Rou=0.9;     %信息素挥发系数
P0=0.2;      %转移概率常数
Lower_1=-1;  %设置搜索范围
Upper_1=1;
Lower_2=-1;
Upper_2=1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:Ant
   X(i,1)=(Lower_1+(Upper_1-Lower_1)*rand);  %随机设置蚂蚁的初始位置
   X(i,2)=(Lower_2+(Upper_2-Lower_2)*rand);
   Tau(i)=F(X(i,1),X(i,2));
end

step=0.05;
f='-(x.^4+3*y.^4-0.2*cos(3*pi*x)-0.4*cos(4*pi*y)+0.6)';


[x,y]=meshgrid(Lower_1:step:Upper_1,Lower_2:step:Upper_2);
z=eval(f);
figure(1);
subplot(1,2,1);
mesh(x,y,z);
hold on;
```




```
plot3(X(:,1),X(:,2),Tau,'k*')
hold on;
text(0.1,0.8,-0.1,'蚂蚁的初始分布位置');
xlabel('x');ylabel('y');zlabel('f(x,y)');

for T=1:Times
   lamda=1/T;
   [Tau_Best(T),BestIndex]=max(Tau);
   for i=1:Ant
      P(T,i)=(Tau(BestIndex)-Tau(i))/Tau(BestIndex);  %计算状态转移概率
   end
   for i=1:Ant
      if P(T,i)<P0  %局部搜索
         temp1=X(i,1)+(2*rand-1)*lamda;
         temp2=X(i,2)+(2*rand-1)*lamda;
      else  %全局搜索
         temp1=X(i,1)+(Upper_1-Lower_1)*(rand-0.5);
         temp2=X(i,2)+(Upper_2-Lower_2)*(rand-0.5);
      end

      %越界处理
      if temp1<Lower_1
         temp1=Lower_1;
      end
      if temp1>Upper_1
         temp1=Upper_1;
      end
      if temp2<Lower_2
         temp2=Lower_2;
      end
      if temp2>Upper_2
         temp2=Upper_2;
      end

      %%%
      if F(temp1,temp2)>F(X(i,1),X(i,2))  %判断蚂蚁是否移动
         X(i,1)=temp1;
         X(i,2)=temp2;
      end
   end
   for i=1:Ant
      Tau(i)=(1-Rou)*Tau(i)+F(X(i,1),X(i,2));  %更新信息量
   end
end

subplot(1,2,2);
mesh(x,y,z);
hold on;
x=X(:,1);
y=X(:,2);
plot3(x,y,eval(f),'k*');
hold on;
text(0.1,0.8,-0.1,'蚂蚁的最终分布位置');
xlabel('x');ylabel('y');zlabel('f(x,y)');

[max_value,max_index]=max(Tau);
maxX=X(max_index,1);
```

```
maxY=X(max_index,2);
maxValue=F(X(max_index,1),X(max_index,2));
```

设定目标函数如下：

```
function [F]=F(x1,x2)
F=-(x1.^4+3*x2.^4-0.2*cos(3*pi*x1)-0.4*cos(4*pi*x2)+0.6);
```

运行程序得到蚂蚁算法运行前后蚂蚁的位置变化示意图，如图 10-13 所示。

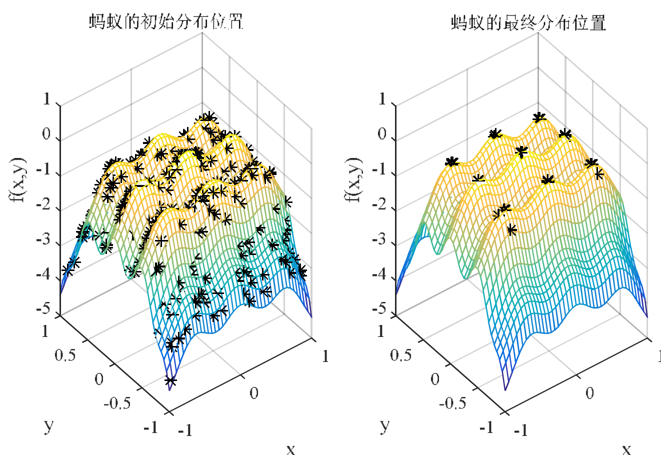


图 10-13　蚂蚁的位置变化示意图

10.3.3　经典应用

移动机器人路径规划是机器人学的一个重要研究领域。它要求机器人依据某个或某些优化原则（如最小能量消耗、最短行走路线、最短行走时间等），在其工作空间中找到一条从起始状态到目标状态能避开障碍物的最优路径。

机器人路径规划问题可以建模为一个有约束的优化问题，都要完成路径规划、定位和避障等任务。

应用蚁群算法求解机器人路径优化问题的主要步骤如下：

（1）输入由 0 和 1 组成的矩阵表示机器人需要寻找最优路径的地图，如图 10-14 所示。

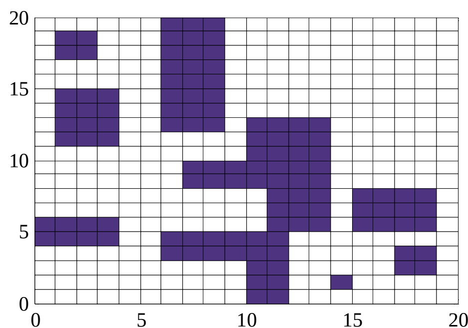


图 10-14　机器人需要寻找最优路径的地图

其中 0 表示此处可以通过，1 表示此处为障碍物。由此得到图 10-14 所表示的矩阵：

```
G=[0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 1 1 0 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0;
   0 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0;
   0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0;
   1 1 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0;
   1 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 0;
   0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0;
   0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0;];
```

（2）输入初始的信息素矩阵，选择初始点和终止点并且设置各种参数。在此次计算中，我们设置所有位置的初始信息素相等。

（3）选择从初始点下一步可以到达的节点，根据每个节点的信息素求出前往每个节点的概率，并利用轮盘算法选取下一步的初始点。

$$p_{ij}^{k}=\begin{cases}\dfrac{\left[\tau_{ij}(t)\right]^{\alpha}\cdot\left[\eta_{ij}\right]^{\beta}}{\sum\limits_{k\in\{N-tabu_k\}}\left[\tau_{ij}(t)\right]^{\alpha}\cdot\left[\eta_{ij}\right]^{\beta}}, & j\in\{N-tabu_k\}\\ 0, & \text{其他}\end{cases}$$

其中$\tau_{ij}(t)$为析取图中弧(i,j)上的信息素的浓度；η_{ij}为与弧(i,j)相关联的启发式信息；α 、β 分别为$\tau_{ij}(t)$、η_{ij}的权重参数。

（4）更新路径及路程长度。

（5）重复步骤 3、4，直到蚂蚁到达终点或者无路可走。

（6）重复步骤 3～5，直到某一代 m 只蚂蚁迭代结束。

（7）更新信息素矩阵，其中没有到达的蚂蚁不计算在内。

$$\tau_{ij}(t+1)=(1-\rho)*\tau_{ij}(t)+\Delta\tau_{ij}$$

$$\Delta\tau_{ij}(t)=\begin{cases}\dfrac{Q}{L_k(t)}, & \text{蚂蚁}k\text{经过}i,\ j\\ 0, & \text{蚂蚁}k\text{不经过}i,\ j\end{cases}$$

（8）重复步骤 3～7，直至第 *n* 代蚂蚁迭代结束。

例 10-5，根据如图 10-14 所示的地图，画出机器人行走的最短路径，并且输入每一轮迭代的最短路径，查看程序的收敛效果。

解：根据以上分析，得到 MATLAB 代码：

```
function main()
G=[0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
   0 1 1 1 0 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0;
   0 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0;
   0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0;
   1 1 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0;
   1 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 0;
   0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0;
   0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0;];
MM=size(G,1);                % G 地形图为 01 矩阵，如果为 1 表示障碍物
Tau=ones(MM*MM,MM*MM);       % Tau 初始信息素矩阵
Tau=8.*Tau;
K=100;                       %迭代次数（指蚂蚁出动多少波）
M=50;                        %蚂蚁个数
S=1 ;                        %最短路径的起始点
E=MM*MM;                     %最短路径的目的点
Alpha=1;                     % Alpha 表征信息素重要程度的参数
Beta=7;                      % Beta 表征启发式因子重要程度的参数
Rho=0.3 ;                    % Rho 信息素蒸发系数
Q=1;                         % Q 信息素增加强度系数
minkl=inf;
mink=0;
minl=0;
D=G2D(G);
N=size(D,1);                 %N 表示问题的规模（像素个数）
 a=1;                        %小方格像素的边长
 Ex=a*(mod(E,MM)-0.5);       %终止点横坐标
 if Ex==-0.5
Ex=MM-0.5;
end
Ey=a*(MM+0.5-ceil(E/MM));        %终止点纵坐标
 Eta=zeros(N);                   %启发式信息，取为至目标点的直线距离的倒数
 %以下启发式信息矩阵
 for i=1:N
 ix=a*(mod(i,MM)-0.5);
   if ix==-0.5
   ix=MM-0.5;
   end
```

```
iy=a*(MM+0.5-ceil(i/MM));
   if i~=E
   Eta(i)=1/((ix-Ex)^2+(iy-Ey)^2)^0.5;
   else
   Eta(i)=100;
   end
end
ROUTES=cell(K,M);          %用细胞结构存储每一代的每一只蚂蚁的爬行路线
PL=zeros(K,M);             %用矩阵存储每一代的每一只蚂蚁的爬行路线长度
                           %启动 K 轮蚂蚁觅食活动，每轮派出 M 只蚂蚁
for k=1:K
for m=1:M
%状态初始化
W=S;                       %当前节点初始化为起始点
Path=S;                    %爬行路线初始化
PLkm=0;                    %爬行路线长度初始化
TABUkm=ones(N);            %禁忌表初始化
TABUkm(S)=0;               %已经在初始点了，因此要排除
DD=D;                      %邻接矩阵初始化
%下一步可以前往的节点
DW=DD(W,:);
DW1=find(DW);
for j=1:length(DW1)
   if TABUkm(DW1(j))==0
      DW(DW1(j))=0;
  end
end
LJD=find(DW);
Len_LJD=length(LJD);       %可选节点的个数
%蚂蚁未遇到食物或者陷入死胡同或者觅食停止
while W~=E&&Len_LJD>=1
%转轮赌法选择下一步怎么走
PP=zeros(Len_LJD);
for i=1:Len_LJD
    PP(i)=(Tau(W,LJD(i))^Alpha)*((Eta(LJD(i)))^Beta);
end
sumpp=sum(PP);
PP=PP/sumpp;%建立概率分布
Pcum(1)=PP(1);
  for i=2:Len_LJD
  Pcum(i)=Pcum(i-1)+PP(i);
  end
Select=find(Pcum>=rand);
to_visit=LJD(Select(1));
%状态更新和记录
Path=[Path,to_visit];               %路径增加
PLkm=PLkm+DD(W,to_visit);           %路径长度增加
W=to_visit;                         %蚂蚁移到下一个节点
   for kk=1:N
      if TABUkm(kk)==0
      DD(W,kk)=0;
      DD(kk,W)=0;
      end
   end
TABUkm(W)=0;                        %已访问过的节点从禁忌表中删除
 DW=DD(W,:);
DW1=find(DW);
```

```
for j=1:length(DW1)
    if TABUkm(DW1(j))==0
       DW(j)=0;
    end
  end
LJD=find(DW);
Len_LJD=length(LJD);%可选节点的个数
 end
%记下每一代每一只蚂蚁的觅食路线和路线长度
 ROUTES{k,m}=Path;
   if Path(end)==E
      PL(k,m)=PLkm;
      if PLkm<minkl
          mink=k;minl=m;minkl=PLkm;
      end
   else
      PL(k,m)=0;
   end
end
%更新信息素
Delta_Tau=zeros(N,N);%更新量初始化
   for m=1:M
     if PL(k,m)
        ROUT=ROUTES{k,m};
        TS=length(ROUT)-1;%跳数
         PL_km=PL(k,m);
        for s=1:TS
          x=ROUT(s);
          y=ROUT(s+1);
          Delta_Tau(x,y)=Delta_Tau(x,y)+Q/PL_km;
          Delta_Tau(y,x)=Delta_Tau(y,x)+Q/PL_km;
        end
     end
  end
Tau=(1-Rho).*Tau+Delta_Tau;%信息素挥发一部分，新增加一部分
 end
%绘图
plotif=1;                                   %是否绘图的控制参数
 if plotif==1                               %绘收敛曲线
    minPL=zeros(K);
   for i=1:K
     PLK=PL(i,:);
     Nonzero=find(PLK);
     PLKPLK=PLK(Nonzero);
     minPL(i)=min(PLKPLK);
   end
figure(1)
plot(minPL);
hold on
grid on
title('收敛曲线变化趋势');
xlabel('迭代次数');
ylabel('最小路径长度'); %绘爬行图
figure(2)
axis([0,MM,0,MM])
for i=1:MM
for j=1:MM
```

```
if G(i,j)==1
x1=j-1;y1=MM-i;
x2=j;y2=MM-i;
x3=j;y3=MM-i+1;
x4=j-1;y4=MM-i+1;
fill([x1,x2,x3,x4],[y1,y2,y3,y4],[0.2,0.2,0.2]);
hold on
else
x1=j-1;y1=MM-i;
x2=j;y2=MM-i;
x3=j;y3=MM-i+1;
x4=j-1;y4=MM-i+1;
fill([x1,x2,x3,x4],[y1,y2,y3,y4],[1,1,1]);
hold on
end
end
end
hold on
title('机器人运动轨迹');
xlabel('坐标 x');
ylabel('坐标 y');
ROUT=ROUTES{mink,minl};
LENROUT=length(ROUT);
Rx=ROUT;
Ry=ROUT;
for ii=1:LENROUT
Rx(ii)=a*(mod(ROUT(ii),MM)-0.5);
if Rx(ii)==-0.5
Rx(ii)=MM-0.5;
end
Ry(ii)=a*(MM+0.5-ceil(ROUT(ii)/MM));
end
plot(Rx,Ry)
end
plotif2=0;%绘各代蚂蚁爬行图
if plotif2==1
figure(3)
axis([0,MM,0,MM])
for i=1:MM
for j=1:MM
if G(i,j)==1
x1=j-1;y1=MM-i;
x2=j;y2=MM-i;
x3=j;y3=MM-i+1;
x4=j-1;y4=MM-i+1;
fill([x1,x2,x3,x4],[y1,y2,y3,y4],[0.2,0.2,0.2]);
hold on
else
x1=j-1;y1=MM-i;
x2=j;y2=MM-i;
x3=j;y3=MM-i+1;
x4=j-1;y4=MM-i+1;
fill([x1,x2,x3,x4],[y1,y2,y3,y4],[1,1,1]);
hold on
end
end
end
```

```
for k=1:K
PLK=PL(k,:);
minPLK=min(PLK);
pos=find(PLK==minPLK);
m=pos(1);
ROUT=ROUTES{k,m};
LENROUT=length(ROUT);
Rx=ROUT;
Ry=ROUT;
for ii=1:LENROUT
Rx(ii)=a*(mod(ROUT(ii),MM)-0.5);
if Rx(ii)==-0.5
Rx(ii)=MM-0.5;
end
Ry(ii)=a*(MM+0.5-ceil(ROUT(ii)/MM));
end
plot(Rx,Ry)
hold on
end
end
function D=G2D(G)
l=size(G,1);
D=zeros(l*l,l*l);
for i=1:l
    for j=1:l
        if G(i,j)==0
            for m=1:l
                for n=1:l
                    if G(m,n)==0
                        im=abs(i-m);jn=abs(j-n);
                        if im+jn==1||(im==1&&jn==1)
                        D((i-1)*l+j,(m-1)*l+n)=(im+jn)^0.5;
                        end
                    end
                end
            end
        end
    end
end
```

运行以上代码，得到收敛曲线（最小路径）变化趋势，如图 10-15 所示。从图 10-15 中可以看出，在大约迭代 40 代时，最小路径长度基本稳定在 38 左右。

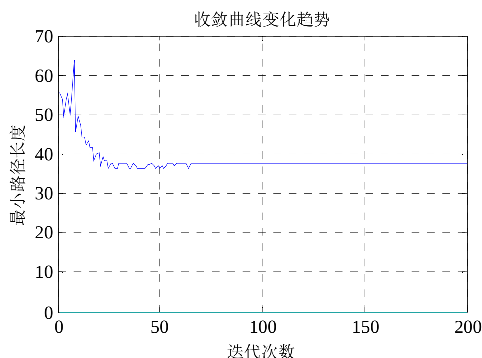


图 10-15　收敛曲线变化趋势

机器人运行轨迹如图 10-16 所示。从图 10-16 可以看出，机器人在到达目标点的整个过程中，成功避开了所有障碍物。

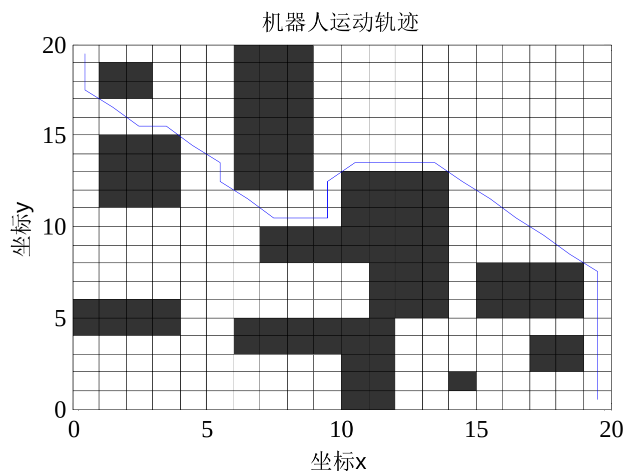


图 10-16　机器人运行轨迹

10.4　本章小结

本章主要介绍了粒子群算法、遗传算法和蚁群算法 3 种常见的经典智能算法，并利用 MATLAB 代码实现其算法过程。最后，通过应用举例详细讲解这 3 种智能算法在 MATLAB 中的应用。

第 3 部分

第 11 章

偏微分方程工具箱



知识要点

本章主要介绍有关解偏微分方程的工具箱。解偏微分方程在数学和物理学中应用广泛，理论丰富。但是，很多理工科的学生，特别是工程人员往往为偏微分方程的复杂求解而挠头。本章的目的是让用户在学会偏微分方程理论以后，能够从容地上手解决几类常见的、实用的偏微分方程，从而提高自己的工作效率。同时，不用考虑底层算法的构建，只需通过 GUI 轻松几步操作就能完成复杂的偏微分方程求解过程。

学习要求	知识点 \ 学习目标	了解	理解	应用	实践
	熟悉偏微分方程工具箱中的常用函数		√		
	利用 GUI 求解椭圆方程			√	√
	利用 GUI 求解抛物线方程			√	√
	利用 GUI 求解双曲线方程			√	√
	利用 GUI 求解特征值方程			√	√

11.1　偏微分方程（PDE）工具箱介绍

前面章节已经介绍了如何利用 MATLAB 求解常微分方程，本节主要介绍利用 MATLAB 自带的 PDE 工具箱求解偏微分方程。众所周知，解偏微分方程不是一件轻松的事情，但是偏微分方程在自然科学和工程领域应用很广，因此，研究解偏微分方程的方法，以及开发解偏微分方程的工具，是数学和计算机领域中的一项重要工作。

MATLAB 提供了专门用于解二维偏微分方程的工具箱，使用这个工具箱，一方面可以解偏微分方程，另一方面可以学习如何把求解数学问题的过程与方法工程化。

11.1.1　偏微分方程常见类型介绍

一般常用的偏微分方程数值解法主要包括有限差分法、有限元法和有限体积法。有限差分法和有限体积法可以参考其他相关文献资料。

MATLAB 的偏微分方程工具箱名字叫 pdetool，它采用有限元法解偏微分方程。使用这个工具箱可以解椭圆方程、抛物线方程、双曲线方程和特征值方程等。下面分别介绍这 4 类方程及其有限元解法。

1．椭圆方程

$$-\nabla(c\nabla u)+au=f$$

方程在二维平面域Ω上。方程中∇是 Laplace 算子，u 是待解的未知函数，c、a、f 是已知的实值标量函数。

在边界$\partial\Omega$上，方程的边界条件一般可以写成：

第一类边界条件（Dirichlet 条件）：

$$hu=r$$

第二类边界条件（Neumann 条件）：

$$n(c\nabla u)+qu=g$$

在两个偏微分方程构成方程组的情况下，边界条件可以写成：

Dirichlet 条件：

$$h_{11}u_1+h_{12}u_2=r_1$$
$$h_{21}u_1+h_{22}u_2=r_2$$

Neumann 条件：

$$\boldsymbol{n}(c_{11}\nabla u_1)+\boldsymbol{n}(c_{12}\nabla u_2)+q_{11}u_1+q_{12}u_2=g_1$$
$$\boldsymbol{n}(c_{21}\nabla u_1)+\boldsymbol{n}(c_{22}\nabla u_2)+q_{21}u_1+q_{22}u_2=g_2$$

混合条件：

$$\boldsymbol{n}(c_{11}\nabla u_1)+\boldsymbol{n}(c_{12}\nabla u_2)+q_{11}u_1+q_{12}u_2=g_1+h_{11}u$$
$$\boldsymbol{n}(c_{21}\nabla u_1)+\boldsymbol{n}(c_{22}\nabla u_2)+q_{21}u_1+q_{22}u_2=g_2+h_{12}u$$

式中，g、h、q、r 是边界 $\partial\Omega$ 上的复值函数，$\boldsymbol{n}$ 是边界 $\partial\Omega$ 上向外的单位法向量。

以下采用变分原理求出虚功方程。

（1）导出定解问题对应的弱形式。

我们讨论椭圆方程在一般 Neumann 条件下的解。取任意试验函数 $v\in V$，同时乘以方程的两边，并在 Ω 上积分：

$$\int_{\Omega}-(\nabla\cdot(c\nabla u))v+auv]\mathrm{d}x=\int_{\Omega}fv\mathrm{d}x$$

利用 Green 公式及 Neumann 条件（第二类边界条件）可得：

$$-\int_{\Omega}[(c\nabla u)\cdot\nabla v+auv]\mathrm{d}x-\int_{\partial\Omega}(-qu+g)v\mathrm{d}s=\int_{\Omega}fv\mathrm{d}x$$

由 $u\in V$，$\forall v\in V$。

于是，问题的虚功方程为：

$$\int_{\Omega}[(c\nabla u)\cdot\nabla v+auv-fv]\mathrm{d}x-\int_{\partial\Omega}(-qu+g)v\mathrm{d}s=0$$

如果问题是自共轭的，并且满足所谓椭圆条件，那么也可以按最小势能原理导出泛函极小问题，这时它与虚功方程是等价的。

（2）区域剖分。

由于三角形剖分在几何上有很大的灵活性，对边界逼近较好，因此在 PDE Toolbox 中区域一般做三角形网格剖分。一般而言，在做三角形剖分和节点编号时要注意以下几点：

- 单元顶点不能是相邻单元边上的内点。
- 尽量避免出现大的钝角、大的边长。
- 在梯度变化比较剧烈的地方，网格要加密。

（3）单元上插值多项式的选取。

在 PDE Toolbox 中采用的是单元上的线性函数：

$$u_e(x,y)=ax+by+c \qquad (x,y)\in e$$

设节点 p_i 上 u 的值为 u_i，即 $u(x_i,y_i)=u_i$，$i=1,2,\cdots,N_p$，N_p 为节点数。任取元 e，3 个顶点为 p_i、p_j、p_m，记 $e=\overline{p_ip_jp_m}$。它们的顺序是逆时针的。为了使插值函数在这 3 个顶点上分别取值 u_i、u_j、u_m，那么 a、b、c 应满足：

$$ax_i+by_i+c=u_i$$
$$ax_j+by_j+c=u_j$$
$$ax_m+by_m+c=u_m$$

解得 a、b、c，再代入线性函数，可得插值函数：

$$u(x,y)=N_i(x,y)u_i+N_j(x,y)u_j+N_m(x,y)u_m$$

其中，

$$N_i(x,y)=\frac{1}{2\Delta_e}(a_ix+b_iy+c_i)$$

$$a_i=\begin{vmatrix} y_j & 1 \\ y_m & 1 \end{vmatrix}$$

$$b_i=\begin{vmatrix} x_i & 1 \\ x_m & 1 \end{vmatrix}$$

$$c_i=\begin{vmatrix} x_i & y_j \\ x_m & y_m \end{vmatrix}$$

$\Delta_e=\frac{1}{2}\begin{vmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_m & y_m & 1 \end{vmatrix}$为单元三角形 $p_ip_jp_m$ 的面积。同理，$N_j(x,y)$、$N_m(x,y)$ 可由 $N_i(x,y)$ 按 i、j、m 的脚标轮换得到，即 $i\to j\to m\to i$。

引入记号：

$$[N]=[N_i(x,y),N_j(x,y),N_m(x,y)]$$

矩阵：

$$\{u\}_e=[u_i,u_j,u_m]^T$$

这样在单元 e 上，有

$$u_e(x,y)=[N]\{u\}_e$$

同时：

$$\nabla u_e(x,y)=\begin{pmatrix} \frac{\partial u_e}{\partial x} \\ \frac{\partial u_e}{\partial y} \end{pmatrix}=[B]\{u\}_e$$

$$[B]=\begin{pmatrix} \frac{\partial N_i}{\partial x} & \frac{\partial N_j}{\partial x} & \frac{\partial N_m}{\partial x} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_j}{\partial y} & \frac{\partial N_m}{\partial y} \end{pmatrix}$$

其中$[B]$是 2×3 常数矩阵。

（4）单元刚度矩阵、单元荷载向量的形成。

设$\{v_i\},i=1,2,\cdots,N_p$，N_p 为 V 的 N_p 维子空间的基函数，按所剖分的单元将虚功方程改成

$$\sum_{n=1}^{NE}\int_{\partial\gamma_n}(g-qu)v\mathrm{d}s+\sum_{n=1}^{NE}\iint_{e_n}[(c\nabla u)\cdot\nabla v+auv-fv]\mathrm{d}x\mathrm{d}y=0$$

这里，v 的向量形式与 u_e 的向量形式相同，并且

$$\iint_{e_n}(c\nabla u)\cdot\nabla v+auv\mathrm{d}x\mathrm{d}y$$

$$=\iint_{e_n}c([B]\{v\}_e)^T([B]\{u\}_e)+a(\{u\}_e\{v\}_e)^T(([B]\{v\}_e)^T)\mathrm{d}x\mathrm{d}y$$

$$=\{v\}_e^{\ T}K_e\{u\}_e$$

$$\iint_{e_n}fv\mathrm{d}x\mathrm{d}y=\iint_{e_n}([N]\{v\}_e)^T f\mathrm{d}x\mathrm{d}y=\{v\}_e^{\ T}\iint_{e_n}([N])^T f\mathrm{d}x\mathrm{d}y=\{v\}_e^{\ T}F_e$$

$$F_{e3\times1}=\iint_{e_n}([N])^T f\mathrm{d}x\mathrm{d}y$$

$$\int_{\partial\gamma_n}quv\mathrm{d}s=\int_{\partial\gamma_n}(q\{v\}_e)^T u\mathrm{d}s=\{v\}_e^{\ T}\{K\}_e\{v\}_e$$

$$\int_{\partial\gamma_n}gv\mathrm{d}s=\int_{\partial\gamma_n}(g\{v\}_e)^T u\mathrm{d}s=\{v\}_e^{\ T}\{\overline{F}\}_e$$

这里三角单元的梯度和面积是通过命令 pdetrg 来实现的。

（5）总刚度阵和总荷载向量的组装。

由上述单元的求值代到总的虚功方程中，从而得到：

$$\sum_{n=1}^{NE}\int_{\partial\gamma_n}quv\mathrm{d}s+\sum_{n=1}^{NE}\iint_{e_n}[(c\nabla u)\cdot\nabla v+auv-fv]\mathrm{d}x\mathrm{d}y=0$$

$$\sum_{n=1}^{NE}\iint_{e_n}c\nabla u\nabla v+auv\mathrm{d}x\mathrm{d}y=\sum_{n=1}^{NE}\int_{\partial\gamma_n}quv\mathrm{d}s-\sum_{n=1}^{NE}\iint_{e_n}fv\mathrm{d}x\mathrm{d}y$$

$$\sum_{n=1}^{NE}\{v\}_e^{\ T}K_e\{u\}_e=\sum_{n=1}^{NE}\{v\}_e^{\ T}G_e-\sum_{n=1}^{NE}\{v\}_e^{\ T}F_e=\sum_{n=1}^{NE}\{v\}_e^{\ T}(G_e-F_e)$$

$$\sum_{n=1}^{NE}\{v\}_e^{\ T}[K_e\{u\}_e-(G_e-F_e)]_{3\times1}=0$$

$$\sum_{n=1}^{NE}[K_e\{u\}_e-(G_e-F_e)]_{3\times1}=0$$

$$KU=(G-F)$$

$$K_{3NE\times3NE}U_{3NE\times1}=\overline{F}$$

$$\overline{F}=(G-F)\big|_{3NE\times1}$$

（6）约束处理，求解方程组。

对于 Neumann 条件，由于是自然边界条件，边界上不需要满足任何约束条件，因此可

立即得到线性代数方程组：

$$KU=F$$

形成 K 和 F，在 MATLAB 环境下立即可解出节点近似解的向量 $\boldsymbol{u}$。

如果是 Dirichlet 边界条件，还需要对边界节点进行约束处理，然后再解方程组。

2．抛物线方程

$$d(\frac{\partial u}{\partial t})-\nabla(c\nabla u)+au=f$$

下面说明如何将抛物线方程简化成椭圆方程来求解。这是通过 PDE Toolbox 中的函数 parabolic 完成的。考虑抛物线方程的初值为

$$u(x,0)=u_0(x), x\in\Omega$$

边界条件类似椭圆边值问题，这里仅讨论 Neumann 条件，即 $\vec{\boldsymbol{n}}(c\nabla u)+qu=g$ 的情况。

抛物线方程可以写成：

$$\rho C\frac{\partial u}{\partial t}-\nabla\cdot(k\nabla u)+h(u-u_\infty)=f$$

表示热量向环境扩散，其中 ρ 是密度，C 是比热，k 是导热系数，h 是薄层传热系数，u_∞ 是环境温度，f 是热源。

如果系数与时间无关，则方程就是标准的椭圆方程：

$$-\nabla(c\nabla u)+au=f$$

对区域 Ω 做三角形网格剖分，对于任意给定的 $t\geqslant 0$，PDE 的解按有限元法的基底可以展开成

$$u(x,t)=\sum_i u_i(t)\varphi_j(x)$$

将展开式代入椭圆标准方程，两边乘以试验函数 φ_j，然后在 Ω 上积分，利用 Green 公式和边界条件，可得，

$$\sum_i\int_\Omega\varphi_j\varphi_j\mathrm{d}x\frac{\mathrm{d}u_i(t)}{\mathrm{d}t}+\sum_i\left(\int_\Omega(c\nabla\varphi_j)\cdot\nabla\varphi_j+a\varphi_j\varphi_j\mathrm{d}x+\int_{\partial\Omega}q\varphi_j\varphi_j\mathrm{d}s\right)v_i(t)\mathrm{d}x$$
$$=\int_\Omega f\varphi_j\mathrm{d}x+\int_{\partial\Omega}g\varphi_j\mathrm{d}s$$

上式可以写成大型线性稀疏的常微分方程组：

$$\boldsymbol{M}\frac{\mathrm{d}u}{\mathrm{d}t}+\boldsymbol{KU}=F$$

这就是所谓的线性半离散化方法。再解上式的初值问题，初值为

$$U_i(0)=u_0(x_i)$$

可得每一个节点 x_i 、任一时刻 t 的 ODE 解。这里 $\boldsymbol{K}$ 和 $\boldsymbol{F}$ 是原边界条件下椭圆方程的刚度矩阵和荷载向量，$\boldsymbol{M}$ 是质量矩阵。

当边界条件是与时间有关的 Dirichlet 条件时，F 则为包括 h 和 r 的时间导数，它可以用有限差分求解。常微分方程组是病态的，这时需做显式时间积分。

由于稳定性要求时间间隔很小，而隐式解由于每一时间段都要求解椭圆方程，从而求解非常缓慢。常微分方程组的数值积分可以由 MATLAB 中的 suite 函数完成。对于这类问题它是有效果的。

3．双曲线方程

$$d(\frac{\partial^2 u}{\partial t^2}) - \nabla(c\nabla u) + au = f$$

类似于解抛物线方程的有限元法，考虑上式的初值为

$$u(x,0) = u_0(x), x \in \Omega$$

$$\frac{\partial u}{\partial t}(x,0) = v_0(x)$$

边界条件同上。

对区域 Ω 做三角剖分，与抛物线方程处理方法一样，可以得到二阶常微分方程组：

$$\boldsymbol{M}\frac{\mathrm{d}^2 V}{\mathrm{d}t^2} + \boldsymbol{K}V = \boldsymbol{F}$$

初值为 $V_i(0) = u_0(x_i), \frac{\mathrm{d}}{\mathrm{d}t}V_i(0) = V_0(x_i)$ 。

其中，$\boldsymbol{K}$ 是刚度矩阵，$\boldsymbol{M}$ 是质量矩阵。

PDE 提供的求解双曲线方程函数命令是 hyperbolic。

4．特征值方程

$$-\nabla(c\nabla u) + au = \lambda du$$

在固体力学中方程描述薄膜振动的问题，在量子力学中应用也很广泛。数值解包括方程的离散和代数特征值问题的求解。首先考虑离散化。按有限元基底将 u 展开，两边同乘基函数，再在区域上做积分，可以得到广义特征值方程：

$$\boldsymbol{KU} = \lambda \boldsymbol{MU}$$

其中对应于右边项的质量矩阵的元素为

$$\boldsymbol{M}_{i,j} = \int_\Omega d(x)\varphi_j(x)\varphi_j(x)\mathrm{d}x$$

在通常情况下，当函数 $d(x)$ 为正时，质量矩阵 $\boldsymbol{M}$ 为正定对称矩阵。同样，当 $c(x)$ 是正的而且在 Dirichlet 边界条件下时，刚度矩阵 $\boldsymbol{K}$ 也是正定的。

对广义特征值问题：

$$KU = l\ MU$$

利用 Arnoldi 算法进行移位和求逆矩阵，直到所有的特征值都落在用户事先确定的区间。在此，求解的详细过程就不再讨论了。

PDE Toolbox 中提供的求解特征值问题的命令函数是 pdeeig。

11.1.2　偏微分方程工具箱简介

在命令行窗口中输入 pdetool，窗口打开进入工作状态。提供两种解方程的方法：一种是通过函数，利用函数可以编程，也可以用命令行的方式解方程，函数及功能如表 11-1 所示；另一种是对窗口进行交互操作。

表 11-1　偏微分方程常用函数表

函　数	功　能
adaptmesh	生成自适应网格及偏微分方程的解
assemb	生成边界质量和刚度矩阵
assema	生成积分区域上的质量和刚度矩阵
assempde	组成偏微分方程的刚度矩阵
hyperbolic	求解双曲线型偏微分方程
parabolic	求解抛物线型偏微分方程
pdeeig	求解特征型偏微分方程
pdenonlin	求解非线性微分方程
poisolv	利用矩阵格式快速求解泊松方程
pdeellip	画椭圆
pdecirc	画圆
pdepoly	画多边形
pderect	画矩形
csgchk	检查几何矩阵的有效性
initmesh	产生最初的三角形网格
pdemesh	画偏微分方程的三角形网格
pdesurf	画表面图命令

一般来说，用函数解方程比较烦琐，而通过窗口交互操作比较简单。解方程的全部过程及结果都可以输出保存为文本文件，限于本文的篇幅，我们主要介绍交互操作解偏微分方程的方法。

1．确定待解的偏微分方程

使用函数 assempde 可以对待解的偏微分方程加以描述。在交互操作中，为了方便用户，把常见问题归结为几个类型，可以在窗口的工具栏上找到选择类型的弹出菜单。这些类型如下：

- 通用问题。
- 通用系统（二维的偏微分方程组）。

- 平面应力。
- 结构力学平面应变。
- 静电学。
- 静磁学。
- 交流电电磁学。
- 直流电导电介质。
- 热传导。
- 扩散。

确定问题类型后，可以在 PDE Specification 对话窗口中输入 c、a、f、d 等系数（函数），这样就确定了待解的偏微分方程。

2．确定边界条件

使用函数 assemb 可以描述边界条件。用 pdetool 提供的边界条件对话框，在对话框里输入 *g*、*h*、*q*、*r* 等边界条件。

3．确定偏微分方程所在域的几何图形

可以用表 11-1 中的函数画出W域的几何图形，如 pdeellip：画椭圆；pderect：画矩形；pdepoly：画多边形。也可以用鼠标在 pdetool 的画图窗中直接画出W域的几何图形。pdetool 提供了类似于函数那样画圆、椭圆、矩形、多边形的工具。

无论哪种画法，图形一经画出，pdetool 就为这个图形自动取名，并把代表图形的名字放入 Set formula 窗口。在这个窗口中，可以实现对图形的拓扑运算，以便构造复杂的W域几何图形。

4．划分有限元

对域进行有限元划分的函数有 initmesh（基本划分）和 refinemesh（精细划分）等。

在 pdetool 窗口中直接单击划分有限元的按钮划分有限元，划分的方法与上面的函数相对应。

5．解方程

经过前 4 步之后就可以解方程了。解方程的函数有，adaptmesh：解方程的通用函数；poisolv：矩形有限元解椭圆型方程；parabolic：解抛物线型方程；hyperbolic：解双曲线型方程。

在 pdetool 窗口中直接单击解方程的按钮即可解方程。解方程所耗费的时间取决于有限元划分的多少。

11.2　求解偏微分方程示例

前面主要介绍了有关 pedtool 工具箱的基本知识，本节介绍关于求解偏微分方程具体示例的操作。

11.2.1　求解椭圆方程

在偏微分方程中，有类特殊的椭圆曲线方程即泊松方程，下面介绍一个这类问题的示例供读者参考。

例 11-1，求解在域W上泊松方程$-DU=1$、边界条件¶W上$U=0$的数值解，其中W是一个方块图形。

首先通过 11.1 节学习的知识来确定求解上述偏微分方程的步骤。

[1] 启动 pdetool 界面。在 MATLAB 命令行窗口中输入 pdetool，按 Enter 键弹出 PDE Toolbox 对话框，然后画一个单位圆，并单击 ∂Ω 图标，弹出如图 11-1 所示的界面。

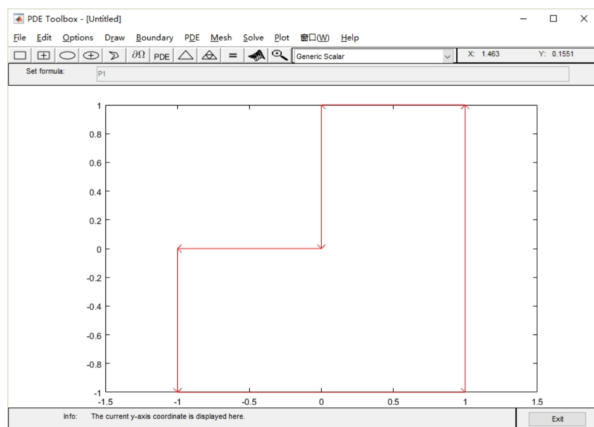


图 11-1　pdetool 界面

[2] 选择 Boundary→Specify boundary conditions 命令，选中 Dirichlet 边界条件按钮并设置 h=1，r=0，如图 11-2 所示。

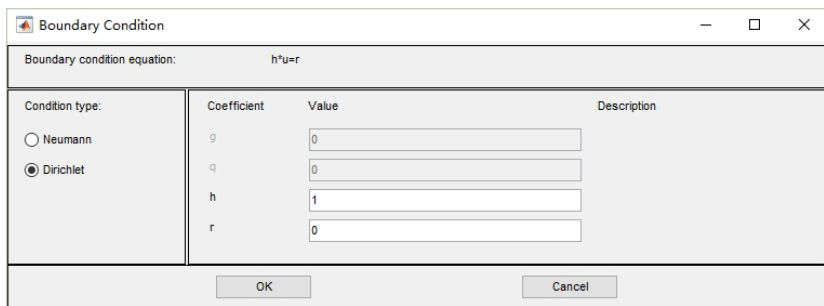


图 11-2　边界条件设置界面

[3] 单击 PDE 按钮，将会弹出如图 11-3 所示的对话框。选中 Elliptic（椭圆）按钮并设置 c=10，a=0.0，f=10.0。

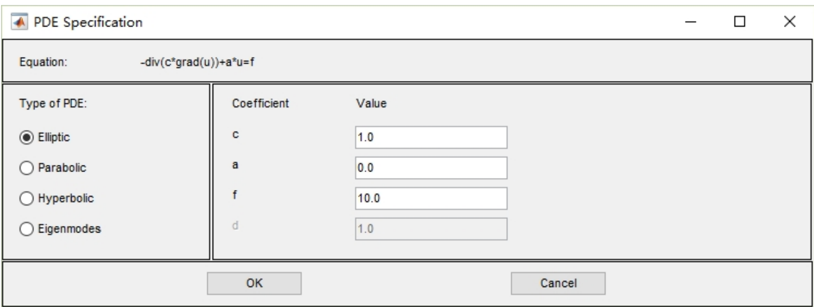


图 11-3　设置偏微分方程类型

[4] 划分单元。单击三角形按钮△，弹出如图 11-4 所示的对话框。继续单击双三角形按钮，弹出如图 11-5 所示的对话框。

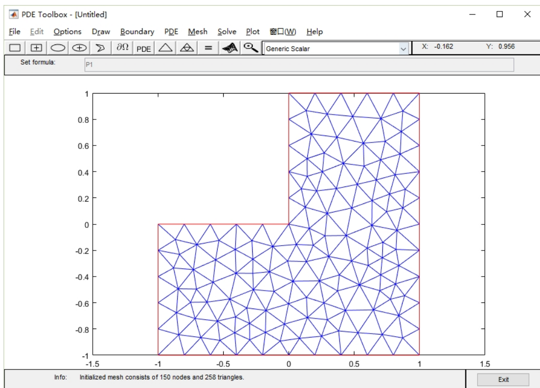


图 11-4　划分三角网格

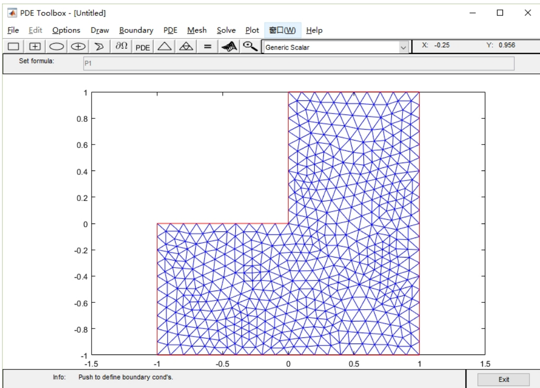


图 11-5　细分三角网格

[5] 求解方程。单击等号按钮，弹出如图 11-6 所示的对话框，显示出求解方程值的分布。

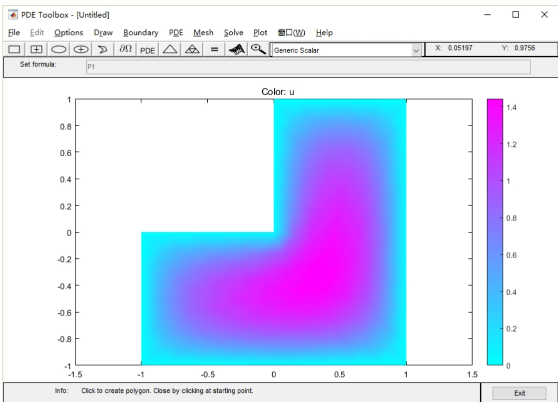


图 11-6　泊松方程的数值解

[6] 对比精确解的绝对误差值。选择 Plot→Parameters 命令，弹出选择框，如图 11-7 所示，在 Property 下选择 User entry 选项，并在其中输入方程的精确解 u-(1-x.^2-y.^2)/4，单击 Plot 按钮，弹出如图 11-8 所示的绝对误差图。

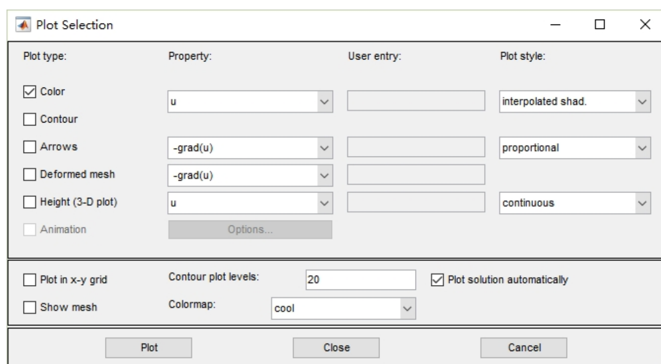


图 11-7　选择框

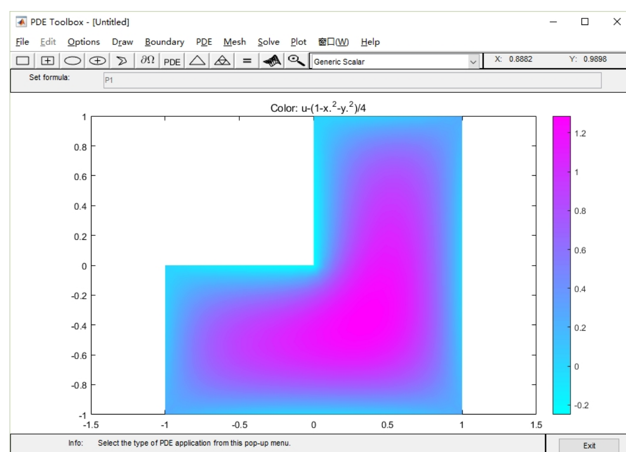


图 11-8　绝对误差图

选择 File→Save as 命令，选择一个文件存放路径。最后，将结果保存为 M 文件即 ell.m，代码如下所示，以后运行此例时在 MATLAB 命令行窗口中执行即可。

```
    function pdemodel
    [pde_fig,ax]=pdeinit;
    pdetool('appl_cb',1);
    set(ax,'DataAspectRatio',[1 1 1]);
    set(ax,'PlotBoxAspectRatio',[921.59999999999991               614.39999999999998
614.39999999999998]);
    set(ax,'XLimMode','auto');
    set(ax,'YLim',[-1.5 1.5]);
    set(ax,'XTickMode','auto');
    set(ax,'YTickMode','auto');
    pdetool('gridon','on');
    % Geometry description:
    pdeellip(-0.00081433224755711464,-0.0024429967426706778,0.99429967426710109,1.00
1628664495114,...
    0,'E1');
    set(findobj(get(pde_fig,'Children'),'Tag','PDEEval'),'String','E1')
    % Boundary conditions:
    pdetool('changemode',0)
    pdesetbd(4,...
    'dir',...
    1,...
    '1',...
```

```
'0')
pdesetbd(3,...
'dir',...
1,...
'1',...
'0')
pdesetbd(2,...
'dir',...
1,...
'1',...
'0')
pdesetbd(1,...
'dir',...
1,...
'1',...
'0')

% Mesh generation:
setappdata(pde_fig,'Hgrad',1.3);
setappdata(pde_fig,'refinemethod','regular');
setappdata(pde_fig,'jiggle',char('on','mean',''));
pdetool('initmesh')
pdetool('refine')

% PDE coefficients:
pdeseteq(1,...
'1.0',...
'0.0',...
'1.0',...
'1.0',...
'0: 10',...
'0.0',...
'0.0',...
'[0 100]')
setappdata(pde_fig,'currparam',...
['1.0';...
'0.0';...
'1.0';...
'1.0'])

% Solve parameters:
setappdata(pde_fig,'solveparam',...
str2mat('0','1620','10','pdeadworst',...
'0.5','longest','0','1E-4','','fixed','Inf'))

% Plotflags and user data strings:
setappdata(pde_fig,'plotflags',[4 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 0 1]);
setappdata(pde_fig,'colstring','u-(1-x.^2-y.^2)/4');
setappdata(pde_fig,'arrowstring','');
setappdata(pde_fig,'deformstring','');
setappdata(pde_fig,'heightstring','');

% Solve PDE:
pdetool('solve')
```

11.2.2　求解抛物线方程

下面向用户介绍一类特殊的抛物线方程，即热方程，请用户参考以下示例。

例 11-2，求在一个矩形域 Ω 上的热方程 $d\frac{\partial u}{\partial t}-\Delta u=0$。边界条件为：在左边界上 $u=100$，在右边界上 $\frac{\partial u}{\partial n}=-10$，在其他边界上 $\frac{\partial u}{\partial n}=0$，其中 Ω 是一个矩形 R1 与矩形 R2 的差，R1：$[-0.5,0.5]\times[-0.8,0.8]$，R2：$[-0.05,0.05]\times[-0.4,0.4]$。

具体解方程步骤如下：

[1] 启动 pdetool 界面。在 MATLAB 命令行窗口中输入 pdetool，按 Enter 键弹出 PDE Toolbox 对话框。选择 Options→Application→Generic Scalar 命令，如图 11-9 所示，然后选中 Grid 之后再选中 Grid Spacing 修改网格尺度，如图 11-10 所示。

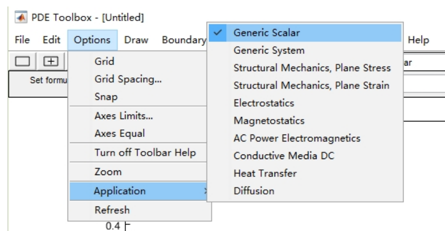


图 11-9　Generic Scalar

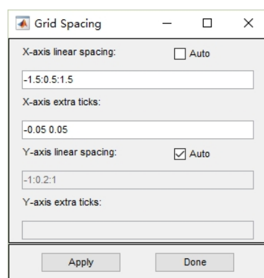


图 11-10　Grid Spacing

[2] 画矩形区域。选择 Draw→Rectangle/square 命令，画 R1 和 R2，并且分别单击坐标系中的 R1 与 R2 图标设置其大小，具体内容如图 11-11 和图 11-12 所示。最后在 Set formula 文本框中输入 R1-R2。

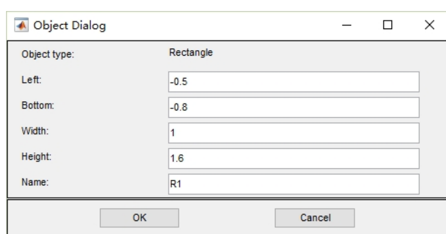


图 11-11　设置 R1

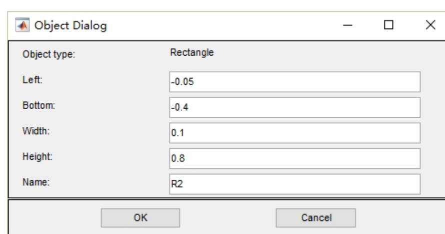


图 11-12　设置 R2

[3] 边界条件。选择 Edit→Select All 命令，并选择 Boundary→Specify boundary conditions 命令，选中 Neumann 边界条件按钮，设置 $\frac{\partial u}{\partial n}=0$。然后分别单击最左侧边界和最右侧边界，按照要求设置边界条件。

[4] 设置方程类型。由于热方程是特殊的抛物线方程，所以选中 Parabolic 单选按钮并设置 c=1.0，a=0.0，f=0，d=1.0，如图 11-13 所示。

[5] 设置时间。选择 Solve→Solve Parameters 命令，然后设置时间，并将 u(t0)设置成 0.0，

其他不变，如图 11-14 所示。

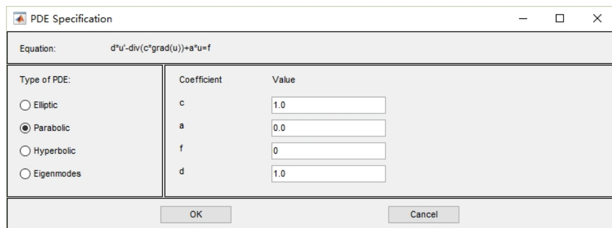


图 11-13　设置偏微分方程类型

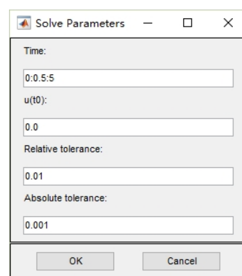


图 11-14　Solve Parameters

[6] 求解热方程。单击等号按钮，弹出如图 11-15 所示的对话框，显示出求解方程值的分布。

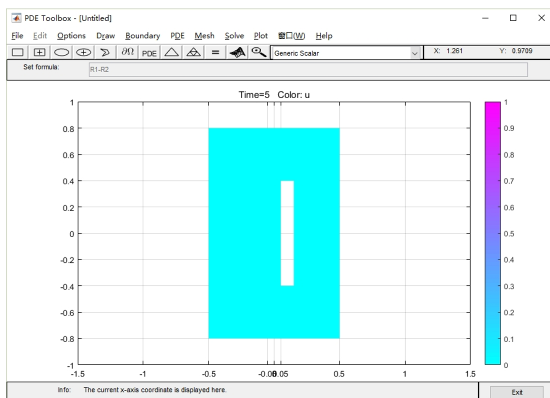


图 11-15　热方程数值解的分布

选择 File→Save as 命令，选择一个文件存放路径。最后，将结果保存为 M 文件，即 par.m，代码如下所示，以后运行此例时在 MATLAB 命令行窗口中执行即可。

```
function pdemodel
[pde_fig,ax]=pdeinit;
pdetool('appl_cb',1);
pdetool('snapon','on');
set(ax,'DataAspectRatio',[1 9.25 1]);
set(ax,'PlotBoxAspectRatio',[1 0.66666666666666663 1]);
set(ax,'XLim',[-1 1]);
set(ax,'YLim',[-1.5 1.5]);
set(ax,'XTick',[ -1.5,...
 -1,...
 -0.5,...
 -0.050000000000000003,...
 0,...
 0.050000000000000003,...
 0.5,...
 1,...
 1.5,...
]);
set(ax,'YTick',[ -1,...
 -0.80000000000000004,...
 -0.59999999999999998,...
```

```
     -0.39999999999999991,...
     -0.19999999999999996,...
     0,...
     0.19999999999999996,...
     0.39999999999999991,...
     0.59999999999999998,...
     0.80000000000000004,...
     1,...
    ]);
    setappdata(ax,'extraspacex','-0.05 0.05');
    pdetool('gridon','on');

    % Geometry description:
    pderect([-0.5 0.5 0.80000000000000004 -0.80000000000000004],'R1');
    pderect([-0.050000000000000003      0.050000000000000003      0.40000000000000002
-0.40000000000000002],'R2');
    set(findobj(get(pde_fig,'Children'),'Tag','PDEEval'),'String','R1-R2')

    % Boundary conditions:
    pdetool('changemode',0)
    pdesetbd(8,...
    'neu',...
    1,...
    '0',...
    '0')
    pdesetbd(7,...
    'neu',...
    1,...
    '0',...
    '0')
    pdesetbd(6,...
    'dir',...
    1,...
    '1',...
    '100')
    pdesetbd(5,...
    'neu',...
    1,...
    '0',...
    '0')
    pdesetbd(4,...
    'neu',...
    1,...
    '0',...
    '0')
    pdesetbd(3,...
    'neu',...
    1,...
    '0',...
    '0')
    pdesetbd(2,...
    'neu',...
    1,...
    '0',...
    '0')
    pdesetbd(1,...
    'neu',...
```

```
    1,...
    '0',...
    '-10')

    % Mesh generation：
    setappdata(pde_fig,'Hgrad',1.3);
    setappdata(pde_fig,'refinemethod','regular');
    setappdata(pde_fig,'jiggle',char('on','mean',''));
    pdetool('initmesh')
    pdetool('refine')

    % PDE coefficients：
    pdeseteq(2,...
    '1.0',...
    '0.0',...
    '0',...
    '1.0',...
    '0:0.5:5',...
    '0.0',...
    '0.0',...
    '[0 100]')
    setappdata(pde_fig,'currparam',...
    ['1.0';...
    '0.0';...
    '0  ';...
    '1.0'])

    % Solve parameters：
    setappdata(pde_fig,'solveparam',...
    str2mat('0','1308','10','pdeadworst',...
    '0.5','longest','0','1E-4','','fixed','Inf'))

    % Plotflags and user data strings：
    setappdata(pde_fig,'plotflags',[1 1 1 1 1 1 1 1 0 0 0 11 1 0 0 0 0 1]);
    setappdata(pde_fig,'colstring','');
    setappdata(pde_fig,'arrowstring','');
    setappdata(pde_fig,'deformstring','');
    setappdata(pde_fig,'heightstring','');

    % Solve PDE：
    pdetool('solve')
```

11.2.3　求解双曲线方程

下面介绍一类特殊的双曲线方程，即波动方程，请用户参考以下示例。

例 11-3，求解在矩形域内的波动方程 $\frac{\partial^2 u}{\partial t^2} - \Delta u = 0$，在左、右边界 $u = 0$，在上、下边界 $\frac{\partial u}{\partial n} = 0$，另外，要求有初值 $u(t_0)$ 与 $\frac{\partial u(t_0)}{\partial t}$，这里从 t=0 开始，那么 $u(0) = a\tan(\cos(\frac{\pi}{2}x))$，从而 $\frac{\partial u(0)}{\partial t} = 3\sin(\pi x)e^{\sin(\frac{\pi}{2}y)}$。

具体解方程步骤如下：

[1] 启动 pdetool 界面。在 MATLAB 命令行窗口中输入 pdetool，按 Enter 键弹出 PDE Toolbox 对话框。然后选择 Options→Generic Scalar 命令。

[2] 画矩形区域。选择 Draw→Rectangle/square 命令，画 *R*1：(-1, -1)，(-1,1)，(1,-1)，(1,1)。

[3] 边界条件。分别选中上、下边界，选择 Boundary→Boundary Mode→Specify Boundary Conditions 命令，选中 Neumann 边界条件，按照要求设置边界条件，如图 11-16 所示。然后分别单击左边界和右边界，按照要求设置边界条件，如图 11-17 所示。

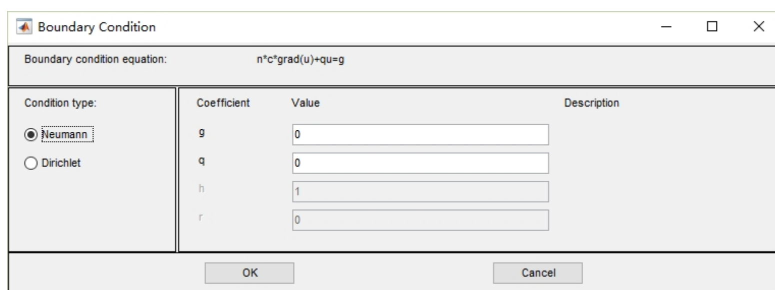


图 11-16　Neumann 条件

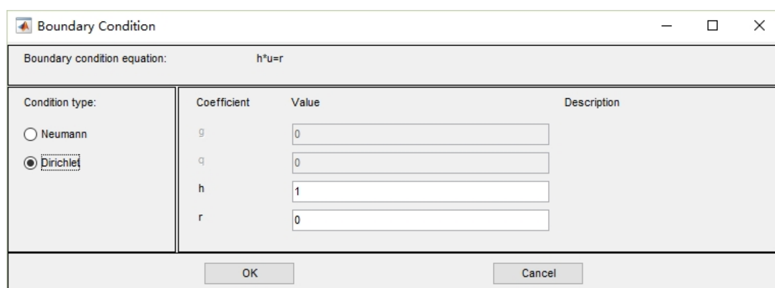


图 11-17　Dirichlet 条件

[4] 设置方程类型。由于波动方程是特殊的双曲线方程，所以选中 Hyperbolic 单选按钮并设置 c=1.0，a=0.0，f=0.0，d=1.0，如图 11-18 所示。

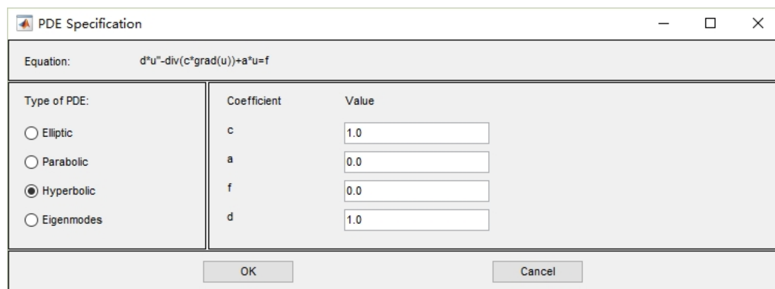


图 11-18　设置偏微分方程类型

[5] 设置时间参数。选择 Solve→Solve Parameters 命令，然后在 Time 文本框中输入 linspace (0,5,31)，在 u(t0)文本框中输入 atan(cos(pi/2*x))，在 u'(t0)文本框中输入 3*sin(pi*x). *exp(sin(pi/2*y))，其他不变，如图 11-19 所示。

[6] 动画效果图。选择 Plot→Parameters 命令，弹出如图 11-20 所示的对话框，选中 Animation 复选框，然后单击 Options 按钮，弹出如图 11-21 所示的对话框，选中 Replay movie 复选框，单击 OK 按钮后，弹出一个动态图像，如图 11-22 所示。

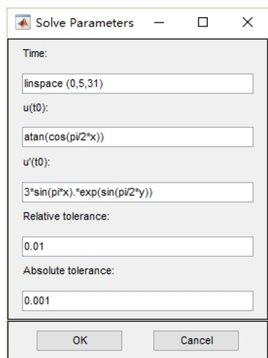


图 11-19　Solve Parameters

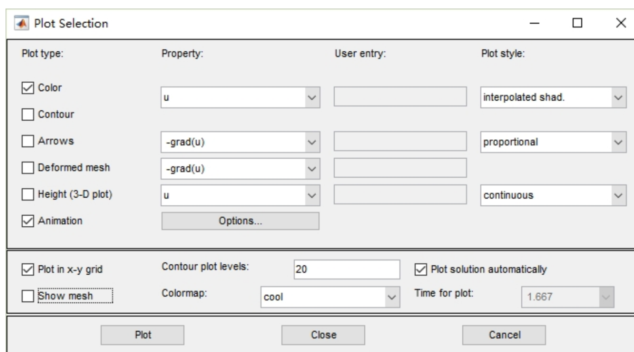


图 11-20　Plot Selection

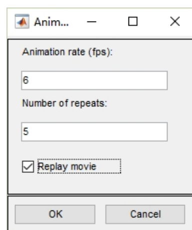


图 11-21　Animation Options

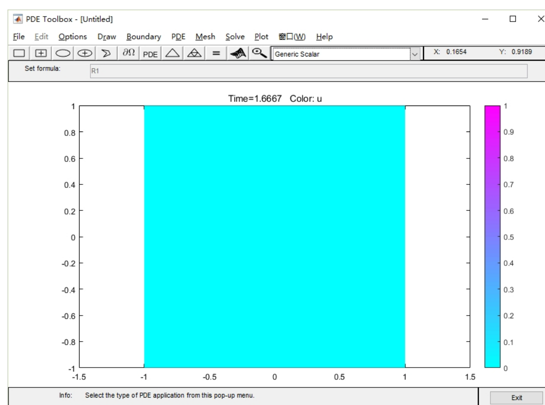


图 11-22　动态图像

[7] 求波动方程。在图 11-20 所示的对话框中取消选中 Animation 复选框，然后单击 Plot 按钮，弹出如图 11-23 所示的对话框，显示出求解方程值的分布。

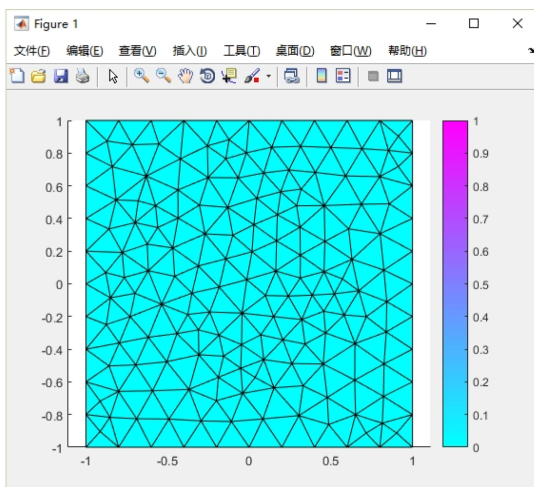


图 11-23　波动方程数值解的分布

选择 File→Save as 命令，选择一个文件存放路径。最后，将结果保存为 M 文件，即 hyp.m，代码如下所示，以后运行此例时在 MATLAB 命令行窗口中执行即可。

```
function pdemodel
[pde_fig,ax]=pdeinit;
pdetool('appl_cb',1);
set(ax,'DataAspectRatio',[1 1.5 1]);
set(ax,'PlotBoxAspectRatio',[1.5 1 1]);
set(ax,'XLim',[-1.5 1.5]);
set(ax,'YLim',[-1.5 1.5]);
set(ax,'XTickMode','auto');
set(ax,'YTick',[ -1.2,...
 -1,...
 -0.79999999999999993,...
 -0.59999999999999987,...
 -0.39999999999999991,...
 -0.19999999999999996,...
 0,...
 0.19999999999999996,...
 0.39999999999999991,...
 0.59999999999999987,...
 0.79999999999999993,...
 1,...
 1.2,...
]);

% Geometry description:
pderect([-1 1 1 -1],'R1');
set(findobj(get(pde_fig,'Children'),'Tag','PDEEval'),'String','R1')

% Boundary conditions:
pdetool('changemode',0)
pdesetbd(4,...
'dir',...
1,...
'1',...
'0')
pdesetbd(3,...
'neu',...
1,...
'0',...
'0')
pdesetbd(2,...
'dir',...
1,...
'1',...
'0')
pdesetbd(1,...
'neu',...
1,...
'0',...
'0')

% Mesh generation:
setappdata(pde_fig,'Hgrad',1.3);
setappdata(pde_fig,'refinemethod','regular');
setappdata(pde_fig,'jiggle',char('on','mean',''));
```

```
pdetool('initmesh')

% PDE coefficients:
pdeseteq(3,...
'1.0',...
'0.0',...
'0.0',...
'1.0',...
'linspace(0,5,31)',...
'atan(cos(pi/2*x))',...
'3*sin(pi*x).*exp(sin(pi/2*y))',...
'[0 100]')
setappdata(pde_fig,'currparam',...
['1.0';...
'0.0';...
'0.0';...
'1.0'])

% Solve parameters:
setappdata(pde_fig,'solveparam',...
str2mat('0','1000','10','pdeadworst',...
'0.5','longest','0','1E-4','','fixed','Inf'))

% Plotflags and user data strings:
setappdata(pde_fig,'plotflags',[1 1 1 1 1 1 1 1 0 1 31 1 0 0 0 0 1]);
setappdata(pde_fig,'colstring','');
setappdata(pde_fig,'arrowstring','');
setappdata(pde_fig,'deformstring','');
setappdata(pde_fig,'heightstring','');

% Solve PDE:
pdetool('solve')
```

11.2.4　求解特征值方程

下面介绍特征值方程的求解过程，请用户参考以下示例。

例 11-4，计算特征值小于 100 的特征方程

$$-\Delta u = \lambda u$$

其中求解区域在 L 形上，拐角点分别是(0,0)，(-1,0)，(-1,-1)，(1,-1)，(1,1)和(0,1)，并且边界条件为 u=0。

具体解方程步骤如下：

[1] 启动 pdetool 界面。在 MATLAB 命令行窗口中输入 pdetool，按 Enter 键弹出 PDE Toolbox 对话框。然后选择 Options→Generic Scalar 命令。

[2] 画 L 多边形区域。选择 Draw→Rectangle/square 命令，画 R1 与 R2：(0,0)，(-1,0)，(-1,-1)，(1,-1)，(1,1)和(0,1)，如图 11-24 所示。

[3] 边界条件。选择 Boundary→Specify Boundary Conditions 命令，选中 Dirichlet 边界条件按钮，按照要求设置边界条件，如图 11-25 所示。

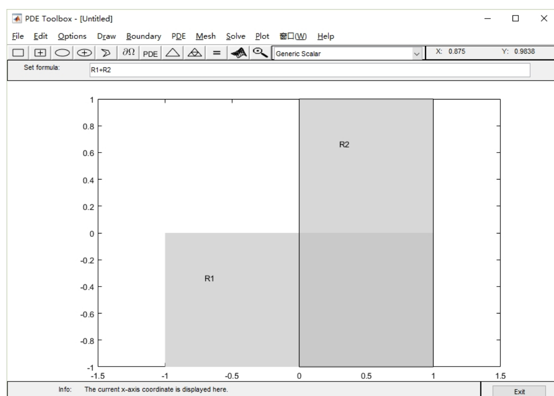


图 11-24　L 形区域

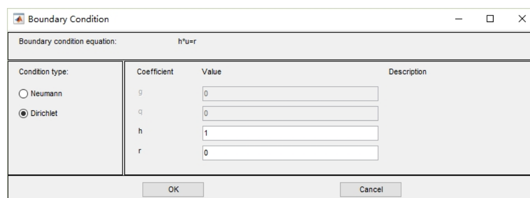


图 11-25　边界条件设置

[4] 设置方程类型。在 PDE Specification 对话框中选择 Eigenmodes 单选按钮，并设置 c=1.0，a=0.0，d=1.0，如图 11-26 所示。

[5] 设置特征值范围。选择 Solve→Parameters 命令，然后输入[0 100]，如图 11-27 所示。

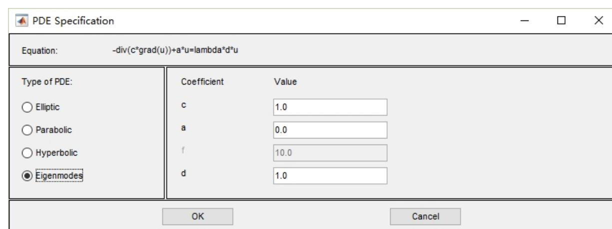


图 11-26　方程类型设置

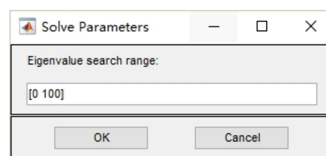


图 11-27　范围设置

[6] 求解特征方程。单击等号按钮，弹出如图 11-28 所示的对话框，显示出求解方程值的分布。

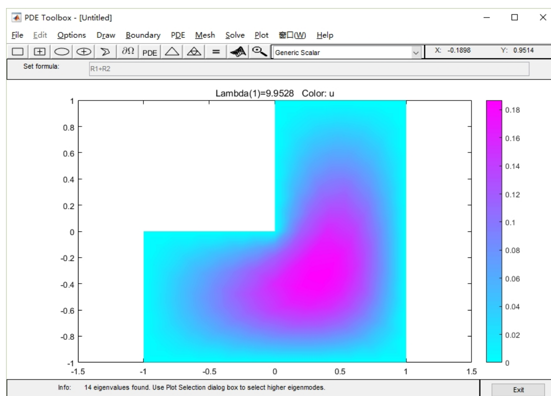


图 11-28　特征方程的解

选择 File→Save as 命令，选择一个文件存放路径。最后，将结果保存为 M 文件，即 Eig.m，代码如下所示，以后运行此例时在 MATLAB 命令行窗口中执行即可。

```
function pdemodel
[pde_fig,ax]=pdeinit;
pdetool('appl_cb',1);
set(ax,'DataAspectRatio',[1 1.2 1]);
```

```
set(ax,'PlotBoxAspectRatio',[1.5 1 1]);
set(ax,'XLim',[-1.5 1.5]);
set(ax,'YLim',[-1 1.3999999999999999]);
set(ax,'XTickMode','auto');
set(ax,'YTickMode','auto');

% Geometry description:
pderect([-1 1 0 -1],'R1');
pderect([0 1 1 -0],'R2');
set(findobj(get(pde_fig,'Children'),'Tag','PDEEval'),'String','R1+R2')

% Boundary conditions:
pdetool('changemode',0)
pdesetbd(7,...
'dir',...
1,...
'1',...
'0')
pdesetbd(6,...
'dir',...
1,...
'1',...
'0')
pdesetbd(5,...
'dir',...
1,...
'1',...
'0')
pdesetbd(4,...
'dir',...
1,...
'1',...
'0')
pdesetbd(3,...
'dir',...
1,...
'1',...
'0')
pdesetbd(2,...
'dir',...
1,...
'1',...
'0')
pdesetbd(1,...
'dir',...
1,...
'1',...
'0')

% Mesh generation:
setappdata(pde_fig,'Hgrad',1.3);
setappdata(pde_fig,'refinemethod','regular');
setappdata(pde_fig,'jiggle',char('on','mean',''));
pdetool('initmesh')

% PDE coefficients:
pdeseteq(4,...
```

```
    '1.0',...
    '0.0',...
    '10.0',...
    '1.0',...
    '0: 10',...
    '0.0',...
    '0.0',...
    '[0 100]')
    setappdata(pde_fig,'currparam',...
    ['1.0 ';...
    '0.0 ';...
    '10.0';...
    '1.0 '])

    % Solve parameters:
    setappdata(pde_fig,'solveparam',...
    str2mat('0','1000','10','pdeadworst',...
    '0.5','longest','0','1E-4','','fixed','Inf'))

    % Plotflags and user data strings:
    setappdata(pde_fig,'plotflags',[1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 0 1]);
    setappdata(pde_fig,'colstring','');
    setappdata(pde_fig,'arrowstring','');
    setappdata(pde_fig,'deformstring','');
    setappdata(pde_fig,'heightstring','');

    % Solve PDE:
    pdetool('solve')
```

用户可以依照以上所举示例，根据自己的不同需求修改边界条件，求解区域和方程参数。当然，可视化操作虽然简单方便，但并不灵活。用户也可以根据自己的能力通过数值计算方法（如有限差分法）来编写自己的代码，还可以选择工具箱中的 File→Save as 命令保存成 M 文件，用户可以自己参看代码。

以上针对 4 类偏微方程（即椭圆、抛物线、特征值及双曲线方程）分别讲解了如何通过 pdetool 工具箱来解决具体的问题，感兴趣的用户也可以参看 help 文档来学习和了解如何解特征方程及偏微分方程组，由于篇幅所限，这里就不再一一赘述了。

11.3 本章小结

本章主要针对工程人员，特别是涉及偏微分方程的数值解法及工程应用而编写的。本章的主要思想是通过具体、简单的示例来概括偏微分方程的数值结果，从结构安排上分别介绍了 4 类常见的偏微分方程在交互式界面上的操作方法，并且将这些操作进一步转化为 MATLAB 语言。用户在学习完本章以后，通过不断地对各种类型的偏微分方程的学习与积累，可以顺利地使用 MATLAB 提供的偏微分工具箱来解决实际问题。

由于篇幅和水平有限，还有一些偏微分方程（组）如非线性方程的解法，用户可以参考 help 文档。感兴趣的用户也可以利用一些常见的求解偏微分方程的数值解法，亲自将其进行算法描述并编写代码，然后在 MATLAB 上运行。

第 12 章

优化工具箱

知识要点

最优化方法就是专门研究如何从多个方案中合理地提取出最佳方案的科学。利用 MATLAB 的优化工具箱，可以求解线性规划、非线性规划和多目标规划问题。另外，该工具箱还提供了线性、非线性最小化，方程求解，曲线拟合，二次规划等问题中大型课题的求解方法，为优化方法在工程中的实际应用提供了更方便、快捷的途径。

学习要求	学习目标 / 知识点	了解	理解	应用	实践
	优化工具箱中的常用函数	√			
	求解最小化问题			√	√
	求解线性规划问题			√	√
	求解无约束非线性规划问题		√		√
	求解二次规划问题			√	√
	求解有约束最小化问题			√	√
	求解目标规划问题			√	√
	求解最大最小化问题			√	√



12.1 优化工具箱及最优化问题介绍

在生活中，人们对于同一个问题往往会提出多个解决方案，并通过各方面的论证从中提取最佳方案。最优化方法就是专门研究如何从多个方案中科学合理地提取出最佳方案的方法。由于优化问题无所不在，目前最优化方法的应用和研究已经深入到了生产和科研的各个领域，如土木工程、机械工程、化学工程、运输调度、生产控制、经济规划、经济管理等，并取得了显著的经济效益和社会效益。

用最优化方法解决最优化问题的技术称为最优化技术，它包含两个方面的内容。

（1）建立数学模型：即用数学语言来描述最优化问题。模型中的数学关系式反映了最优化问题所要达到的目标和各种约束条件。

（2）数学求解：数学模型建好以后，选择合理的最优化方法进行求解。

下面介绍有关优化工具箱中的常用函数。

12.1.1 优化工具箱常用函数

首先介绍几个优化工具箱的常用函数。利用 optimset 函数，可以创建和编辑参数结构；利用 optimget 函数，可以获得 options 优化参数。

1．optimset 函数

optimset 函数用于创建或编辑优化选项参数结构。其调用格式如下：

```
options = optimset('param1',value1,'param2',value2,...)
optimset
options = optimset
options = optimset(optimfun)
options = optimset(oldopts,'param1',value1,...)
options = optimset(oldopts,newopts)
```

描述如下：

- options=optimset('param1',value1,'param2',value2,...) 的作用是创建一个名为 options 的优化选项参数，其中指定的参数具有指定值。所有未指定的参数都设置为空矩阵[]（将参数设置为[]表示当 options 传递给优化函数时给参数赋默认值）。赋值时只要输入参数前面的字母就可以了。
- optimset 函数没有输入/输出变量时，将显示一张完整的带有有效值的参数列表。
- options = optimset 的作用是创建一个选项结构 options，其中所有的元素被设置为[]。
- options=optimset(optimfun) 的作用是创建一个含有所有参数名和与优化函数 optimfun 相关的带有默认值的选项结构 options。
- options=optimset(oldopts,'param1',value1,...) 的作用是创建一个 oldopts 的备份，用指定的数值修改参数。
- options=optimset(oldopts,newopts) 的作用是将已经存在的选项结构 oldopts 与新的选项结构 newopts 进行合并。newopts 参数中的所有元素将覆盖 oldopts 参数中的所

有对应元素。

2．optimget 函数

optimget 函数用于获取优化选项参数值。其调用格式如下：

```
val = optimget(options,'param')
val = optimget(options,'param',default)
```

描述如下：

- val = optimget(options,'param')：返回指定的参数 param 的值。
- val = optimget(options,'param',default)：返回指定的参数 param 的值，如果该值没有定义则返回默认值。

举例如下：

（1）下面的语句创建一个名为 options 的优化选项结构，其中显示参数设置为 iter，TolFun 参数设置为 1e－8。

```
>> options = optimset('Display','iter','TolFun',1e-8)
```

结果显示：

```
options =
                   Display:  'iter'
               MaxFunEvals:  []
                   MaxIter:  []
                    TolFun:  1.0000e-008
                      TolX:  []
               FunValCheck:  []
                 OutputFcn:  []
                  PlotFcns:  []
           ActiveConstrTol:  []
                 Algorithm:  []
    AlwaysHonorConstraints:  []
            BranchStrategy:  []
           DerivativeCheck:  []
               Diagnostics:  []
             DiffMaxChange:  []
             DiffMinChange:  []
               FinDiffType:  []
         GoalsExactAchieve:  []
                GradConstr:  []
                   GradObj:  []
                   HessFcn:  []
                   Hessian:  []
                  HessMult:  []
               HessPattern:  []
                HessUpdate:  []
           InitialHessType:  []
         InitialHessMatrix:  []
          InitBarrierParam:  []
     InitTrustRegionRadius:  []
                  Jacobian:  []
                 JacobMult:  []
              JacobPattern:  []
                LargeScale:  []
        LevenbergMarquardt:  []
```

```
                LineSearchType:  []
                      MaxNodes:  []
                    MaxPCGIter:  []
                 MaxProjCGIter:  []
                    MaxRLPIter:  []
                    MaxSQPIter:  []
                       MaxTime:  []
                 MeritFunction:  []
                     MinAbsMax:  []
           NodeDisplayInterval:  []
            NodeSearchStrategy:  []
              NonlEqnAlgorithm:  []
            NoStopIfFlatInfeas:  []
                ObjectiveLimit:  []
         PhaseOneTotalScaling:  []
                Preconditioner:  []
              PrecondBandWidth:  []
                RelLineSrchBnd:  []
       RelLineSrchBndDuration:  []
                 ScaleProblem:  []
                       Simplex:  []
           SubproblemAlgorithm:  []
                        TolCon:  []
                     TolConSQP:  []
                    TolGradCon:  []
                        TolPCG:  []
                     TolProjCG:  []
                  TolProjCGAbs:  []
                     TolRLPFun:  []
                   TolXInteger:  []
                      TypicalX:  []
                   UseParallel:  []
```

（2）下面的语句创建一个名为 options 的优化结构的备份，用于改变 TolX 参数的值，将新值保存到 optnew 参数中。

```
optnew = optimset(options,'TolX',1e-4)
```

（3）下面的语句返回 options 优化结构，其中包含所有的参数名和与 fminbnd 函数相关的默认值。

```
options = optimset('fminbnd');
```

（4）若只希望看到 fminbnd 函数的默认值，只需要简单地输入下面的语句就可以了。

```
optimset fminbnd 或 optimset('fminbnd')
```

（5）可以使用下面的命令获取 TolX 参数的值：

```
Tol=optimget(options, 'TolX')
```

得到：

```
Tol=1.0000e-04
```

下面列出有关最优化的 MATLAB 函数，包括最小化函数和方程求解函数，详细描述如表 12-1 和表 12-2 所示。

表 12-1　最小化函数表

函　数	描　述	函　数	描　述
fgoalattain	多目标达到问题	fminsearch, fminunc	无约束非线性最小化
fminbnd	有边界的标量非线性最小化	fseminf	半无限问题
fmincon	有约束的非线性最小化	linprog	线性课题
fminimax	最大最小化	quadprog	二次课题

表 12-2　方程求解函数表

函　数	描　述
solve	线性方程求解
fsolve	非线性方程求解
fzero	标量非线性方程求解

使用优化工具箱时，由于优化函数要求目标函数和约束条件满足一定的格式，所以需要用户在进行模型输入时注意以下几个问题：

（1）目标函数最小化。优化函数 fminbnd、fminsearch、fminunc、fmincon、fgoalattain、fminmax 和 lsqnonlin 都要求目标函数最小化，如果优化问题要求目标函数最大化，可以通过使该目标函数的负值最小化，即-f(x)最小化来实现。同理，对于 quadprog 函数提供-H 和-f，对于 linprog 函数提供-f。

（2）约束非正。优化工具箱要求非线性不等式约束的形式为 Ci(x)≤0，通过对不等式取负可以达到使大于零的约束形式变为小于零的不等式约束形式的目的。如 Ci(x)≥0 形式的约束等价于-Ci(x)≤0；Ci(x)≥b 形式的约束等价于-Ci(x)+b≤0。

12.1.2　最优化问题

求解单变量最优化问题的方法有很多种，根据目标函数是否需要求导可以分为两类，即直接法和间接法。直接法不需要对目标函数进行求导，而间接法则需要用到目标函数的导数。

1．直接法

常用的一维直接法主要有消去法和多项式近似法两种。

（1）消去法。该法利用单峰函数具有的消去性质进行反复迭代，逐渐消去不包含极小点的区间，缩小搜索区间，直到搜索区间缩小到给定的允许精度为止。一种典型的消去法为黄金分割法（Golden Section Search）。黄金分割法的基本思想是在单峰区间内适当插入两点，将区间分为三段，然后通过比较这两点函数值的大小来确定是删去最左段还是最右段，或同时删去左右两段保留中间段。重复该过程使区间无限缩小。插入点的位置放在区间的黄金分割点及其对称点上，所以该法称为黄金分割法。该法的优点是算法简单、效率较高、稳定性好。

（2）多项式近似法。该法用于目标函数比较复杂的情况。此时寻找一个与它近似的函数代替目标函数，并用近似函数的极小点作为原函数极小点的近似。常用的近似函数为二次

和三次多项式。

2．间接法

间接法需要计算目标函数的导数，优点是计算速度很快。常见的间接法包括牛顿切线法、对分法、割线法和三次插值多项式近似法等。优化工具箱中用得较多的是三次插值法。

对于只需要计算函数值的方法，二次插值法是一个很好的方法，它的收敛速度较快，尤其在极小点所在区间较小时更是如此。黄金分割法则是一种十分稳定的方法，并且计算简单。由于以上原因，MATLAB 优化工具箱中用得较多的方法是二次插值法，三次插值法，二次、三次混合插值法和黄金分割法。

下面介绍有关函数。

fminbnd：功能为找到固定区间内单变量函数的最小值。其调用格式如下：

```
x = fminbnd(fun,x1,x2)
x = fminbnd(fun,x1,x2,options)
x = fminbnd(fun,x1,x2,options,P1,P2,...)
[x,fval] = fminbnd(...)
[x,fval,exitflag] = fminbnd(...)
[x,fval,exitflag,output] = fminbnd(...)
```

描述如下：

- x = fminbnd(fun,x1,x2)：返回区间($x1$,$x2$)上 fun 参数描述的标量函数的最小值 x。
- x = fminbnd(fun,x1,x2,options)：用 options 参数指定的优化参数进行最小化。
- x = fminbnd(fun,x1,x2,options,P1,P2,...)：提供另外的参数 $P1$、$P2$ 等，传输给目标函数 fun。如果没有设置 options 选项，则令 options=[]。
- [x,fval] = fminbnd(...)：返回解 x 处目标函数的值。
- [x,fval,exitflag] = fminbnd(...)：返回 exitflag 值描述 fminbnd 函数的退出条件。
- [x,fval,exitflag,output] = fminbnd(...)：返回包含优化信息的结构输出。

与 fminbnd 函数相关的细节内容包含在 fun、options、exitflag 和 output 等参数中，如表 12-3 所示。

表 12-3　参数描述表

参　数	描　述
fun	需要最小化的目标函数。fun 函数需要输入标量参数 x，返回 x 处的目标函数标量值 f。可以将 fun 函数指定为命令行，如： `x = fminbnd(inline('sin(x*x)'),x0)` 同样，fun 参数可以是一个包含函数名的字符串。对应的函数可以是 M 文件、内部函数或 MEX 文件。若 fun='myfun'，则 M 文件函数 myfun.m 必须是下面的形式： `function f = myfun(x)` `f = ...`
options	优化参数选项。可以用 optimset 函数设置或改变这些参数的值。options 参数介绍如下： – Display：显示的内容。选择 off，不显示输出；选择 iter，显示每一步迭代过程的输出；选择 final，显示最终结果。 – MaxFunEvals：函数评价的最大允许次数。 – MaxIter：最大允许迭代次数。

（续表）

参　数	描　述
	– TolX ：*x* 处的终止容限。
exitflag	描述退出条件如下： – >0：表示目标函数收敛于解 *x* 处。 – =0：表示已经达到函数评价或迭代的最大次数。 – <0：表示目标函数不收敛。
output	该参数包含的优化信息如下： – utput.iterations：迭代次数。 – output.algorithm：所采用的算法。 – output.funcCount：函数评价次数。

下面举几个求最小化问题的示例。

例 12-1，在区间(0,2π)上求函数 cos(*x*)的最小值。

```
x = fminbnd(@cos,0,2*pi)
x =
   10.1416
y = cos(x)
```

所以区间(0,2π)上函数 cos(*x*)的最小值点位于 *x*=10.1416 处。

最小值处的函数值为：

```
y =
   -1
```

例 12-2，对边长为 3m 的正方形铁板，在 4 个角处剪去相等的正方形以制成方形无盖水槽，问如何剪才能使水槽的容积最大？

现在要求在区间 (0,1.5)上确定一个 *x*，使容积最大化。因为优化工具箱中要求目标函数最小化，所以需要对目标函数进行转换，即要求最小化。

首先编写 M 文件 opt1.m：

```
function f = opt1(x)
f = -(3-2*x).^2 * x;
```

然后调用 fminbnd 函数：

```
x = fminbnd(@opt1,0,1.5)
```

得到问题的解：

```
x =
   0.5000
```

即剪掉的正方形的边长为 0.5m 时水槽的容积最大。

12.2　线性规划

线性规划是处理线性目标函数和线性约束的一种较为成熟的方法，目前已经广泛应用于军事、经济、工业、农业、教育、商业和社会科学等许多方面。

线性规划的标准形式要求目标函数最小化、约束条件取等式、变量非负。不符合条件的

线性模型要首先转化成标准形。

线性规划的求解方法主要是单纯形法（Simple Method），该法由 Dantzig 于 1947 年提出，后经多次改进。单纯形法是一种迭代算法，它从所有基本可行解的一个较小部分通过迭代过程选出最优解。其迭代过程的一般描述如下：

（1）将线性规划问题化为典范形式，从而可以得到一个初始基本可行解 $x^{(0)}$（初始顶点），将它作为迭代过程的出发点，其目标值为 $z(x^{(0)})$。

（2）寻找一个基本可行解 $x^{(1)}$，使 $z(x^{(1)})\leqslant z(x^{(0)})$。方法是通过消去法将产生 $x^{(0)}$的典范形式化为产生 $x^{(1)}$的典范形式。

（3）继续寻找较好的基本可行解 $x^{(2)},x^{(3)},\cdots$，使目标函数值不断改进，即 $z(x^{(1)})\geqslant z(x^{(2)})\geqslant z(x^{(3)})\geqslant\cdots$。当某个基本可行解再也不能被其他基本可行解改进时，它就是所求的最优解。

MATLAB 优化工具箱中采用的是投影法，它是单纯形法的一种变种。

下面介绍有关函数。

12.2.1 线性规划函数

（1）功能：求解线性规划问题。

（2）调用格式如下：

```
x = linprog(f,A,b,Aeq,beq)
x = linprog(f,A,b,Aeq,beq,lb,ub)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)
[x,fval] = linprog(...)
[x,fval,exitflag] = linprog(...)
[x,fval,exitflag,output] = linprog(...)
[x,fval,exitflag,output,lambda] = linprog(...)
```

（3）描述如下：

- x = linprog(f,A,b)：求解问题 min f'x，约束条件为 $Ax\leqslant b$。
- x = linprog(f,A,b,Aeq,beq)：求解上面的问题，但增加等式约束，即 Aeqx = beq。若没有不等式存在，则令 A=[]、b=[]。
- x = linprog(f,A,b,Aeq,beq,lb,ub)：定义设计变量 x 的下界 lb 和上界 ub，使得 x 始终在该范围内。若没有等式约束，令 Aeq=[]、beq=[]。
- x = linprog(f,A,b,Aeq,beq,lb,ub,x0)：设置初值为 x0。该选项只适用于中型问题，默认大型算法将忽略初值。
- x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)：用 options 指定的优化参数进行最小化。
- [x,fval] = linprog(...)：返回解 x 处的目标函数值 fval。
- [x,lambda,exitflag] = linprog(...)：返回 exitflag 值，描述函数计算的退出条件。
- [x,lambda,exitflag,output] = linprog(...)：返回包含优化信息的输出变量 output。
- [x,fval,exitflag,output,lambda] = linprog(...)：将解 x 处的拉格朗日乘子返回到 lambda 参数中。

（4）变量：lambda 参数介绍。

lambda 参数是解 x 处的拉格朗日乘子，它的属性如下：

- lambda.lower：lambda 的下界。
- lambda.upper：lambda 的上界。
- lambda.ineqlin：lambda 的线性不等式。
- lambda.eqlin：lambda 的线性等式。

（5）算法。

- 大型优化算法：采用 LIPSOL 法，该法在进行迭代计算之前首先要进行一系列的预处理。
- 中型优化算法：linprog 函数使用的是投影法，就像 quadprog 函数的算法一样。linprog 函数使用的是一种活动集方法，是线性规划中单纯形法的变种，它通过求解另一个线性规划问题来找到初始可行解。

（6）诊断。

大型优化问题算法的第一步涉及一些约束条件的预处理问题，有些问题可能导致 linprog 函数退出，并显示不可行的信息。

若 Aeq 参数中某行的所有元素都为零，但 Beq 参数中对应的元素不为零，则给出如下退出信息：

```
    Exiting due to infeasibility： an all zero row in the constraint matrix does not have
a zero in corresponding right hand size entry.
```

若 x 的某一个元素没在界内，则给出以下退出信息：

```
    Exiting due to infeasibility:objective f'*x is unbounded below.
```

若 Aeq 参数的某一行中只有一个非零值，则 x 中的相关值称为奇异变量。这里，x 中该成分的值可以用 Aeq 和 Beq 算得。若算得的值与另一个约束条件相矛盾，则给出如下退出信息：

```
    Exiting due to infeasibility:Singleton variables in equality constraints are not
feasible.
```

若奇异变量可以求解，但其解超出上界或下界，则给出如下退出信息：

```
    Exiting due to infeasibility:singleton variables in the equality constraints are not
within bounds.
```

12.2.2　线性规划问题的应用

1．生产决策问题

例 12-3，某厂生产甲、乙两种产品，已知制成每吨产品甲需用资源 A 3 吨，资源 B 4m^3；制成每吨产品乙需用资源 A 2 吨，资源 B 6m^3，资源 C 7 个单位。若每吨产品甲和乙的经济价值分别为 7 万元和 5 万元，3 种资源的限制量分别为 80 吨、220m^3 和 230 个单位，试分析应生产这两种产品各多少吨才能使创造的总经济价值最高？

这里可以令生产产品甲的数量为 x_1，生产产品乙的数量为 x_2。根据题意，代码设置如下：

```
    clc
```

```
clear
f = [-7;-5];
A =[3 2
     4 6
     0 7];
b  = [80; 220; 230];
lb = zeros(2,1);
```

然后调用 linprog 函数：

```
[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb)
```

最优化结果如下：

```
Optimization terminated.
x =
    4.7619
   39.8571
fval =
 -197.6190
exitflag =
     1
output =
         iterations:  6
          algorithm:  'large-scale:  interior point'
       cgiterations:  0
            message:  'Optimization terminated.'
    constrviolation:  0
      firstorderopt:  1.1390e-007
lambda =
    ineqlin:  [3x1 double]
      eqlin:  [0x1 double]
      upper:  [2x1 double]
      lower:  [2x1 double]
```

由上可知，生产甲种产品 4.7619 吨、乙种产品 39.8571 吨可使创造的总经济价值最高，最高经济价值为 197.619 万元。exitflag=1 表示过程正常收敛于解 x 处。

2. 工作人员计划安排问题

例 12-4，某昼夜服务的公共交通系统每天各时间段（每 4 小时为一个时间段）所需的值班人数如表 12-4 所示，这些值班人员在某一时段开始上班后要连续工作 8 小时（包括轮流用餐时间），问该公共交通系统至少需要多少名工作人员才能满足值班的需要？

表 12-4　各时段所需值班人数表

班　次	时　间　段	所 需 人 数
1	6:00—10:00	50
2	10:00—14:00	30
3	14:00—18:00	70
4	18:00—22:00	60
5	22:00—2:00	40
6	2:00—6:00	20

这里可以设 x_i 为第 i 个时段开始上班的人员数。

根据题意，代码设置如下：

```
clc
clear
f = [1;1;1;1;1;1];
A=[-1 0 0 0 0 -1
   -1 -1 0 0 0 0
   0 -1 -1 0 0 0
   0 0 -1 -1 0 0
   0 0 0 -1 -1 0
   0 0 0 0 -1 -1];
b=[-50;-30;-70;-60;-40;-20];
lb = zeros(6,1);
```

然后调用 linprog 函数：

```
[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb)
Optimization terminated.
```

最优化结果如下：

```
Optimization terminated.
x =
   25.7604
   25.0651
   44.9349
   25.8143
   14.1857
   24.2396
fval =
  160.0000
exitflag =
     1
output =
       iterations： 6
        algorithm： 'large-scale： interior point'
     cgiterations： 0
          message： 'Optimization terminated.'
   constrviolation： 10.9933e-012
     firstorderopt： 10.9933e-012
lambda =
   ineqlin： [6x1 double]
     eqlin： [0x1 double]
     upper： [6x1 double]
     lower： [6x1 double]
```

可见，只要 6 个时段分别安排 26 人、25 人、45 人、26 人、14 人和 24 人就可以满足值班的需要，共计 160 人。并且计算结果 exitflag =1 是收敛的。

3．投资问题

例 12-5，某单位有一批资金用于 4 个工程项目的投资，用于各工程项目时所得的净收益（投入资金的百分比）如表 12-5 所示。

表 12-5　工程项目收益表

工 程 项 目	A	B	C	D
收益（%）	18	10	9	12

由于某种原因，决定用于项目 A 的投资不大于其他各项投资之和；而用于项目 B 和 C 的投资要大于项目 D 的投资。试确定使该单位收益最大的投资分配方案。

这里可以用 x_1、x_2、x_3 和 x_4 分别代表用于项目 A、B、C 和 D 的投资百分数，由于各项目的投资百分数之和必须等于 100%，所以

$$x_1+x_2+x_3+x_4=1$$

根据题意，代码设置如下：

```
f = [-0.18;-0.1;-0.09;-0.12];
A =  [1 -1 -1 -1
      0 -1 -1 1];
b = [0; 0];
Aeq=[1 1 1 1];
beq=[1];
lb = zeros(4,1);
```

然后调用 linprog 函数：

```
[x,fval,exitflag,output,lambda] = linprog(f,A,b,Aeq,beq,lb)
```

结果如下：

```
x =
   0.5000
   0.2500
   0.0000
   0.2500
fval =
  -0.1450
exitflag =
    1
output =
       iterations:  9
        algorithm:  'large-scale:  interior point'
     cgiterations:  0
          message:  'Optimization terminated.'
    constrviolation:  4.4409e-016
      firstorderopt:  9.4035e-010
lambda =
   ineqlin:  [2x1 double]
     eqlin:  0.1450
     upper:  [4x1 double]
     lower:  [4x1 double]
```

上面的结果说明，项目 A、B、C、D 投入资金的百分比分别为 50%、25%、0%、25%时，该单位收益最大。

4．工件加工任务分配问题

例 12-6，某车间有两台机床甲和乙，可用于加工 3 种工件。假定这两台机床的可用台时数分别为 600 和 900，3 种工件的数量分别为 400、600 和 500，且已知用两台不同机床加工

单位数量的不同工件所需的台时数和加工费用（如表 12-6 所示），问怎样分配机床的加工任务，才能既满足加工工件的要求，又使总加工费用最低？

表 12-6　机床加工情况表

机床类型	单位工作所需加工台时数			单位工件的加工费用			可用台时数
	工件 1	工件 2	工件 3	工件 1	工件 2	工件 3	
甲	0.6	1.2	1.1	13	9	10	600
乙	0.4	1.2	1	11	12	8	900

这里可以设在甲机床上加工工件 1、2 和 3 的数量分别为 x_1、x_2 和 x_3，在乙机床上加工工件 1、2 和 3 的数量分别为 x_4、x_5 和 x_6。根据 3 种工种的数量限制，则有：

$x_1+x_4=400$　（对工件 1）

$x_2+x_5=600$　（对工件 2）

$x_3+x_6=500$　（对工件 3）

根据题意，代码设置如下：

```
clc
clear
f = [13;9;10;11;12;8];
A =  [0.6 1.2 1.1 0 0 0
      0 0 0 0.4 1.2 1.0];
b = [600; 900];
Aeq=[1 0 0 1 0 0
     0 1 0 0 1 0
     0 0 1 0 0 1];
beq=[400 600 500];
lb = zeros(6,1);
```

然后调用 linprog 函数：

```
[x,fval,exitflag,output,lambda] = linprog(f,A,b,Aeq,beq,lb)
```

结果如下：

```
Optimization terminated.
x =
    0.0000
  500.0000
    0.0000
  400.0000
  100.0000
  500.0000
fval =
  1.4100e+004
exitflag =
     1
output =
         iterations:  5
          algorithm:  'large-scale:  interior point'
       cgiterations:  0
            message:  'Optimization terminated.'
    constrviolation:  5.1159e-013
      firstorderopt:  6.4382e-010
```

```
lambda =
   ineqlin:  [2x1 double]
     eqlin:  [3x1 double]
     upper:  [6x1 double]
     lower:  [6x1 double]
```

可见，在甲机床上加工 500 个工件 2，在乙机床上加工 400 个工件 1、加工 500 个工件 3，可在满足条件的情况下使总加工费用最小。最小费用为 14 100 元，收敛正常。

5．厂址选择问题

例 12-7，A、B、C 三地，每地都出产一定数量的产品，也消耗一定数量的原料（如表 12-7 所示）。已知制成每吨产品需 3 吨原料，各地之间的距离为：A—B，150km；A—C，100km；B—C，200km。假定每万吨原料运输 1km 的运价是 5000 元，每万吨产品运输 1km 的运价是 6000 元。由于地区条件的差异，在不同地点设厂的生产费用也不同。问究竟在哪些地方设厂，规模多大，才能使总费用最小？另外，由于其他条件限制，在 B 处建厂的规模（生产的产品数量）不能超过 6 万吨。

表 12-7　A、B、C 三地出产产品、消耗原料情况表

地　　点	年产原料（万吨）	年销产品（万吨）	生产费用（万元/万吨）
A	21	6	150
B	17	12	120
C	22	0	100

这里可令 x_{ij} 为由 i 地运到 j 地的原料数量（万吨），y_{ij} 为由 i 地运往 j 地的产品数量（万吨），i,j=1,2,3（分别对应 A、B、C 三地）。

根据题意，代码设置如下：

```
clc
clear
f = [75;75;50;50;100;100;150;240;210;120;160;220];
A=[1 -1 1 -1 0 0 3 3 0 0 0 0
  -1 1 0 0 1 -1 0 0 3 3 0 0
   0 0 -1 1 -1 1 0 0 0 0 3 3
   0 0 0 0 0 0 0 0 1 1 0 0];
b=[21;17;22;6];
Aeq=[0 0 0 0 0 0 1 0 1 0 1 0
    0 0 0 0 0 0 0 1 0 1 0 1];
beq=[6;12];
lb = zeros(12,1);
```

然后调用 linprog 函数：

```
[x,fval,exitflag,output,lambda] = linprog(f,A,b,Aeq,beq,lb)
```

结果如下：

```
Optimization terminated.
x =
   0.0000
   0.0000
   0.0000
   0.0000
   0.0000
```

```
      0.0000
      6.0000
      0.0000
      0.0000
      5.6667
      0.0000
      6.3333
fval =
   2.9733e+03
exitflag =
     1
output =
         iterations： 8
          algorithm： 'large-scale： interior point'
       cgiterations： 0
            message： 'Optimization terminated.'
    constrviolation： 7.1054e-015
      firstorderopt： 1.6812e-012
```

可见要使总费用最小，A、B、C 三地的建厂规模分别为 6 万吨、5.666 7 万吨和 6.333 万吨，最小总费用为 2.973 3e+03 万元。

6．确定职工编制问题

例 12-8，某工厂每日 8 小时的产量不低于 1 800 件。为了进行质量控制，计划聘请两个不同水平的检验员。一级检验员的速度为：25 件/小时，正确率 98%，计时工资 4 元/小时；二级检验员的速度为：15 件/小时，正确率 95%，计时工资 3 元/小时。检验员每错检一次，工厂要损失 2 元。现有可供厂方聘请的检验员人数为一级 7 人和二级 8 人。为使总检验费用最省，该工厂应聘一级、二级检验员各多少名？

可以设需要一级和二级检验员的人数分别为 x_1 名和 x_2 名。根据题意，代码设置如下：

```
clc
clear
f = [40;36];
A=[1 0
   0 1
   -5 -3];
b=[7;8;-45];
lb = zeros(2,1);
```

然后调用 linprog 函数：

```
[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb)
```

结果如下：

```
Optimization terminated.
x =
    7.0000
    10.3333
fval =
   400.0000
exitflag =
     1
output =
         iterations： 6
          algorithm： 'large-scale： interior point'
```

```
        cgiterations:  0
             message:  'Optimization terminated.'
     constrviolation:  0
       firstorderopt:  4.9052e-007
lambda =
    ineqlin:  [3x1 double]
      eqlin:  [0x1 double]
      upper:  [2x1 double]
      lower:  [2x1 double]
```

可见，招聘一级检验员 7 名、二级检验员 3 名可使总检验费用最省，约为 400.00 元。计算收敛。

7．生产计划的最优化问题

例 12-9，某工厂生产 A 和 B 两种产品，它们需要经过 3 种设备的加工，其工时如表 12-8 所示。设备一、二和三每天可使用的时间分别不超过 11 小时、9 小时和 12 小时。产品 A 和 B 的利润随市场的需求有所波动，如果预测未来某个时期内 A 和 B 的利润分别为 5 000 元/吨和 3 000 元/吨，问在哪个时期内，每天应生产产品 A、B 各多少吨，才能使工厂获利最大？

表 12-8　生产产品工时表

产　　品	设 备 一	设 备 二	设 备 三
A（小时/吨）	4	5	6
B（小时/吨）	3	4	3
设备每天最多可工作时数（小时）	11	9	12

这里可以设每天应安排生产产品 A 和 B 分别为 x_1 吨和 x_2 吨。

根据题意，代码设置如下：

```
clc
clear
f = [-5;-3];
A=[4 3
   5 4
   6 3];
b=[11;9;12];
lb = zeros(2,1);
```

然后调用 linprog 函数：

```
[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb)
```

结果如下：

```
Optimization terminated.
x =
    1.8000
    0.0000
fval =
   -9.0000
exitflag =
     1
output =
          iterations:  6
```

```
        algorithm:  'large-scale:  interior point'
      cgiterations:  0
         message:  'Optimization terminated.'
    constrviolation:  0
      firstorderopt:  8.2385e-009
lambda =
    ineqlin:  [3x1 double]
      eqlin:  [0x1 double]
      upper:  [2x1 double]
      lower:  [2x1 double]
```

所以，每天生产 A 产品 1.80 吨、B 产品 0 吨可使工厂获得最大利润 9 000 元/吨。

12.3　无约束非线性规划

无约束最优化问题在实际应用中也比较常见，如工程中常见的参数反演问题。另外，许多有约束最优化问题可以转换为无约束最优化问题进行求解。

12.3.1　基本数学原理介绍

求解无约束最优化问题的方法主要有两类，即直接搜索法（Search Method）和梯度法（Gradient Method）。

直接搜索法适用于目标函数高度非线性、没有导数或导数很难计算的情况。由于实际工程中很多问题都是非线性的，直接搜索法不失为一种有效的解决办法。常用的直接搜索法为单纯形法，此外还有 Hooke-Jeeves 搜索法、Pavell 共轭方向法等。

在函数的导数可求的情况下，梯度法是一种更优的方法，该法利用函数的梯度（一阶导数）和 Hessian 矩阵（二阶导数）构造算法，可以获得更快的收敛速度。函数 $f(x)$的负梯度方向-▽$f(x)$反映了函数的最大下降方向。当搜索方向取为负梯度方向时称为最速下降法。常见的梯度法有最速下降法、Newton 法、Marquart 法、共轭梯度法和拟牛顿法（Quasi-Newton Method）等。

在所有这些方法中，用得最多的是拟牛顿法，这个方法在每次迭代过程中建立曲率信息，构成二次模型问题。

下面介绍有关 MATLAB 优化工具箱中的求解无约束最优化问题算法。

- 大型优化算法：若用户在函数中提供梯度信息，则函数默认将选择大型优化算法，该算法是基于内部映射牛顿法的子空间置信域法。计算中的每一次迭代涉及用 PCG 法求解大型线性系统得到的近似解。
- 中型优化算法：fminunc 函数的参数 options.LargeScale 设置为 off。该算法采用的是基于二次和三次混合插值一维搜索法的 BFGS 拟牛顿法。但一般不建议使用最速下降法。
- 默认时的一维搜索算法：当 options.LineSearchType 设置为 quadcubic 时，将采用二次和三次混合插值法。将 options.LineSearchType 设置为 cubicpoly 时，将采用三次

插值法。第二种方法需要的目标函数计算次数更少，但梯度的计算次数更多。这样，如果提供了梯度信息，或者能较容易地算得，则三次插值法是更好的选择。

上述涉及的算法局限性主要表现在以下 4 个方面：

（1）目标函数必须是连续的。fminunc 函数有时会给出局部最优解。

（2）fminunc 函数只对实数进行优化，即 *x* 必须为实数，而且 *f*(*x*)必须返回实数。当 *x* 为复数时，必须将它分解为实部和虚部。

（3）在使用大型算法时，用户必须在 fun 函数中提供梯度（options 参数中 GradObj 属性必须设置为 on）。

（4）目前，若在 fun 函数中提供了解析梯度，则 options 参数 DerivativeCheck 不能用于大型算法以比较解析梯度和有限差分梯度。通过将 options 参数的 MaxIter 属性设置为 0 来用中型方法核对导数，然后重新用大型方法求解问题。

12.3.2　无约束非线性规划函数

1．fminunc 函数

（1）功能：求多变量无约束函数的最小值。

调用格式如下：

- x = fminunc(fun,x0)。
- x = fminunc(fun,x0,options)。
- x = fminunc(fun,x0,options,P1,P2,...)。
- [x,fval] = fminunc(...)。
- [x,fval,exitflag] = fminunc(...)。
- [x,fval,exitflag,output] = fminunc(...)。
- [x,fval,exitflag,output,grad] = fminunc(...)。
- [x,fval,exitflag,output,grad,hessian] = fminunc(...)。

（2）描述如下：

- fminunc 给定初值，求多变量标量函数的最小值，常用于无约束非线性最优化问题。
- x = fminunc(fun,x0)：给定初值 *x*0，求 fun 函数的局部极小点 *x*。*x*0 可以是标量、向量或矩阵。
- x = fminunc(fun,x0,options)：用 options 参数中指定的优化参数进行最小化。
- x = fminunc(fun,x0,options,P1,P2,...)：将问题参数 *p*1、*p*2 等直接传递给目标函数 fun，将 options 参数设置为空矩阵，作为 options 参数的默认值。
- [x,fval] = fminunc(...)：将解 *x* 处目标函数的值返回到 fval 参数中。
- [x,fval,exitflag] = fminunc(...)：返回 exitflag 值，描述函数的输出条件。
- [x,fval,exitflag,output] = fminunc(...)：返回包含优化信息的结构输出。
- [x,fval,exitflag,output,grad] = fminunc(...)：将解 *x* 处 fun 函数的梯度值返回到 grad 参数中。

- [x,fval,exitflag,output,grad,hessian] = fminunc(...)：将解 *x* 处目标函数的 Hessian 矩阵信息返回到 hessian 参数中。

（3）变量。

输入/输出变量的描述如表 12-9 所示。

表 12-9　输入/输出变量描述表

变 量	描　　述
fun	目标函数。需要最小化的目标函数。fun 函数需要输入标量参数 *x*，返回 *x* 处的目标函数标量值 *f*。可以将 fun 函数指定为命令行，如： x = fminbnd(inline('sin(x*x) '),x0) 同样，fun 函数可以是一个包含函数名的字符串。对应的函数可以是 M 文件、内部函数或 MEX 文件。若 fun='myfun'，则 M 文件函数 myfun.m 必须有下面的形式： function f = myfun(x) f = ... 若 fun 函数的梯度可以算得，且 options.GradObj 设为 on： options = optimset('GradObj', 'on') 则 fun 函数必须返回解 *x* 处的梯度向量 *g* 到第二个输出变量中去。当被调用的 fun 函数只需要一个输出变量时（如算法只需要目标函数的值而不需要其梯度值时），可以通过核对 nargout 的值来避免计算梯度值。 function [f,g] = myfun(x) f = ...　　：计算 *x* 处的函数值。 if nargout > 1　　：调用 fun 函数并要求有两个输出变量。 g = ...　　：计算 *x* 处的梯度值。 end 若 Hessian 矩阵也可以求得，并且 options.Hessian 设为 on，即 options = optimset('Hessian', 'on') 则 fun 函数必须返回解 *x* 处的 Hessian 对称矩阵 ***H*** 到第 3 个输出变量中去。当被调用的 fun 函数只需要一个或两个输出变量时（如算法只需要目标函数的值 *f* 和梯度值 *g* 而不需要 Hessian 矩阵 ***H*** 时），可以通过核对 nargout 的值来避免计算 Hessian 矩阵
options	优化参数选项。可以通过 optimset 函数设置或改变这些参数。其中有的参数适用于所有的优化算法，有的则只适用于大型优化问题，另外一些则只适用于中型问题。 首先描述适用于大型问题的选项。这仅仅是一个参考，因为使用大型问题算法有一些条件。对于 fminunc 函数来说，必须提供梯度信息。 – LargeScale：当设为 on 时，使用大型算法；若设为 off，则使用中型问题的算法。 适用于大型和中型算法的参数如下： – Diagnostics：打印最小化函数的诊断信息。 – Display：显示水平。选择 off，不显示输出；选择 iter，显示每一步迭代过程的输出；选择 final，显示最终结果。打印最小化函数的诊断信息。 – GradObj：用户定义的目标函数的梯度。对于大型问题此参数是必选的，对于中型问题则是可选项。 – MaxFunEvals：函数评价的最大次数。 – MaxIter：最大允许迭代次数。 – TolFun：函数值的终止容限。 – TolX：*x* 处的终止容限。 只用于大型算法的参数如下： – Hessian：用户定义的目标函数的 Hessian 矩阵。 – HessPattern：用于有限差分的 Hessian 矩阵的稀疏形式。若不方便求 fun 函数的稀疏 Hessian 矩阵 ***H***，可以通过用梯度的有限差分获得的 ***H*** 的稀疏结构（如非零值的位置等）来得到近似的 Hessian 矩阵 ***H***。若连矩阵的稀疏结构都不知道，则可以将 HessPattern 设为密集矩阵，在每一次迭代过程中，都将进行密集矩阵的有限差分近似（这是默认设置）。这将非常麻烦，所以花一些力气得到 Hessian 矩阵的稀疏结构还是值得的。

（续表）

变 量	描 述
	– MaxPCGIter：PCG 迭代的最大次数。 – PrecondBandWidth：PCG 前处理的上带宽，默认为零。对于有些问题，增加带宽可以减少迭代次数。 – TolPCG：PCG 迭代的终止容限。 – TypicalX：典型 *x* 值。 只用于中型算法的参数如下： – DerivativeCheck：对用户提供的导数和有限差分求出的导数进行对比。 – DiffMaxChange：变量有限差分梯度的最大变化。 – DiffMinChange：变量有限差分梯度的最小变化。 – LineSearchType：一维搜索算法的选择。
exitflag	描述退出条件如下： – >0：表示目标函数收敛于解 *x* 处。 – 0：表示已经达到函数评价或迭代的最大次数。 – <0：表示目标函数不收敛。
output	该参数包含的优化信息如下： – output.iterations：迭代次数。 – output.algorithm：所采用的算法。 – output.funcCount：函数评价次数。 – output.cgiterations：PCG 迭代次数（只适用于大型规划问题）。 – output.stepsize：最终步长的大小（只用于中型问题）。 – output.firstorderopt：一阶优化的度量，解 *x* 处梯度的范数。

2．minsearch 函数

（1）功能：求解多变量无约束函数的最小值。

（2）调用格式如下：

- x = fminsearch(fun,x0)。
- x = fminsearch(fun,x0,options)。
- x = fminsearch(fun,x0,options,P1,P2,...)。
- [x,fval] = fminsearch(...)。
- [x,fval,exitflag] = fminsearch(...)。
- [x,fval,exitflag,output] = fminsearch(...)。

（3）描述如下：

- fminsearch 求解多变量无约束函数的最小值。该函数常用于无约束非线性最优化问题。
- x = fminsearch(fun,x0)：初值为 *x*0，求 fun 函数的局部极小点 *x*。*x*0 可以是标量、向量或矩阵。
- x = fminsearch(fun,x0,options)：用 options 参数指定的优化参数进行最小化。
- x = fminsearch(fun,x0,options,P1,P2,...)：将问题参数 *P*1、*P*2 等直接传递给目标函数 fun，将 options 参数设置为空矩阵，作为 options 参数的默认值。
- [x,fval] = fminsearch(...)：将 *x* 处的目标函数值返回到 fval 参数中。
- [x,fval,exitflag] = fminsearch(...)：返回 exitflag 值，描述函数的退出条件。
- [x,fval,exitflag,output] = fminsearch(...)：返回包含优化信息的输出参数 output。

（4）变量。

各变量的意义同前。

（5）算法。

fminsearch 使用单纯形法进行计算。

对于求解二次以上的问题，fminsearch 函数比 fminunc 函数有效。但是，当问题为高度非线性时，fminsearch 函数更具稳健性。

（6）局限性。

应用 fminsearch 函数可能会得到局部最优解。fminsearch 函数只对实数进行最小化，即 x 必须由实数组成，$f(x)$函数必须返回实数。如果 x 是复数，必须将它分为实部和虚部两部分。

12.3.3　无约束非线性规划问题的应用

例 12-10，求解下列无约束非线性函数的最小值：

$$f(x) = 3x_1^2 + 2x_1x_2 + x_2^2$$

代码设置如下：

```
function f = ex10_10(x)
f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;
```

在 MATLAB 命令行窗口中输入下列代码：

```
x0 = [1,1];
[x,fval] = fminunc(@ex10_10,x0)
```

结果如下：

```
x =
  1.0e-006 *
    0.2541   -0.2029
fval =
  1.3173e-013
```

12.4　二次规划

如果某非线性规划的目标函数为自变量的二次函数，约束条件全是线性函数，就称这种规划为二次规划。

12.4.1　二次规划函数 quadprog

（1）功能：求解二次规划问题。

（2）调用格式如下：

- x = quadprog(H,f,A,b)。
- x = quadprog(H,f,A,b,Aeq,beq)。
- x = quadprog(H,f,A,b,Aeq,beq,lb,ub)。
- x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)。

- x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)。
- [x,fval] = quadprog(...)。
- [x,fval,exitflag] = quadprog(...)。
- [x,fval,exitflag,output] = quadprog(...)。
- [x,fval,exitflag,output,lambda] = quadprog(...)。

（3）描述如下：

- x = quadprog(H,f,A,b)：返回向量 *x*，最小化函数 $1/2x'Hx+f'x$，其约束条件为 $Ax\leqslant b$。
- x = quadprog(H,f,A,b,Aeq,beq)：仍然求解上面的问题，但添加了等式约束条件 Aeqx = beq。
- x = quadprog(H,f,A,b,lb,ub)：定义设计变量的下界 lb 和上界 ub，使得 $lb\leqslant x\leqslant ub$。
- x = quadprog(H,f,A,b,lb,ub,x0)：同上，并设置初值 *x*0。
- x = quadprog(H,f,A,b,lb,ub,x0,options)：根据 options 参数指定的优化参数进行最小化。
- [x,fval] = quadprog(...)：返回解 *x* 处的目标函数值 $fval = 0.5x'Hx + f'x$。
- [x,fval,exitflag] = quadprog(...)：返回 exitflag 参数，描述计算的退出条件。
- [x,fval,exitflag,output] = quadprog(...)：返回包含优化信息的结构输出 output。
- [x,fval,exitflag,output,lambda] = quadprog(...)：返回解 *x* 处包含拉格朗日乘子的 lambda 参数。

注意：

①如果问题不是严格凸性的，用 quadprog 函数得到的可能是局部最优解。

②如果用 Aeq 和 Beq 明确地指定等式约束，而不是用 lb 和 ub 指定，则可以得到更好的数值解。

③若 *x* 的组分没有上限或下限，则 quadprog 函数希望将对应的组分设置为 Inf（对于上限）或-Inf（对于下限），而不是强制性地给予上限一个很大的正数或给予下限一个很小的负数。

④对于大型优化问题，若没有提供初值 *x*0，或 *x*0 不是严格可行的，则 quadprog 函数会选择一个新的初始可行点。

⑤若为等式约束，且 quadprog 函数发现负曲度（Negative Curvature），则优化过程终止，exitflag 的值等于-1。

（4）算法。

- 大型优化算法：当优化问题只有上界和下界，而没有线性不等式或等式约束时，则默认算法为大型算法。或者，如果优化问题中只有线性等式，而没有上界和下界或线性不等式时，默认算法也是大型算法。

该算法是基于内部映射牛顿法（Interior-reflective Newton Method）的子空间置信域法（Subspace Trust-region），每一次迭代都与用 PCG 法求解大型线性系统得到的近似解有关。

- 中型优化算法：quadprog 函数使用活动集法，它也是一种投影法，首先通过求解线性规划问题来获得初始可行解。

（5）诊断。

① 大型优化问题。大型优化问题不允许约束上限和下限相等，若 lb(2)==ub(2)，则给出如下出错信息：

```
    Equal upper and lower bounds not permitted in this large-scale method.Use equality
constraints and the medium-scale method instead.
```

若优化模型中只有等式约束，仍然可以使用大型算法；如果模型中既有等式约束又有边界约束，则必须使用中型方法。

② 中型优化问题。当解不可行时，quadprog 函数给出以下警告：

```
    Warning:The constraints are overly stringent;there is no feasible solution.
```

这里，quadprog 函数生成一个结果，这个结果使得约束矛盾最小。

当等式约束不连续时，给出下面的警告信息：

```
    Warning:The equality constraints are overly stringent;there is no feasible solution.
```

当 Hessian 矩阵为负半定时，则生成无边界解，给出下面的警告信息：

```
    Warning:The solution is unbounded and at infinity;the constraints are not restrictive
enough.
```

这里，quadprog 函数返回满足约束条件的 x 值。

（6）局限性。

显示水平只能选择 off 和 final，迭代参数 iter 不可用。当问题不定或负定时，常常无解（此时 exitflag 参数给出一个负值，表示优化过程不收敛）。若正定解存在，则 quadprog 函数可能只给出局部极小值，因为问题可能是非凸的。对于大型问题，不能依靠线性等式，因为 Aeq 必须是行满秩的，即 Aeq 的行数必须不多于列数。若不满足要求，必须调用中型算法进行计算。

12.4.2　二次规划问题的应用

例 12-11，找到使函数最小化的 x 值。

$$f(x)=\frac{1}{2}x_1^2x_2^2-x_1x_2-2x_1-6x_2$$

其中，

$$\begin{cases} x_1+x_2\leqslant 2 \\ -x_1+2x_2\leqslant 2 \\ 2x_1+x_2\leqslant 3 \\ 0\leqslant x_1,0\leqslant x_2 \end{cases}$$

代码设置如下：

```
clc
clear
H = [1 -1; -1 2];
f = [-2; -6];
A = [1 1; -1 2; 2 1];
b = [2; 2; 3];
lb = zeros(2,1);
```

```
%调用二次规划函数
[x,fval,exitflag,output,lambda] = quadprog(H,f,A,b,[],[],lb)
```

最优化结果如下：

```
Optimization terminated.
x =
    0.6667
    1.3333
fval =
   -8.2222
exitflag =
     1
output =
         iterations:  3
    constrviolation:  1.1102e-016
          algorithm:  'medium-scale:  active-set'
      firstorderopt:  8.8818e-016
       cgiterations:  []
            message:  'Optimization terminated.'
lambda =
      lower:  [2x1 double]
      upper:  [2x1 double]
      eqlin:  [0x1 double]
    ineqlin:  [3x1 double]
```

可见 x_1=0.667、x_2=1.333 时 $f(x)$最小，并且方程求解一致收敛。

12.5　有约束最小化

在有约束最优化问题中，通常要将该问题转换为更简单的子问题，这些子问题可以求解并作为迭代过程的基础。

早期的方法通常是通过构造惩罚函数来将有约束最优化问题转换为无约束最优化问题进行求解。现在，这些方法已经被更有效的基于 K-T（Kuhn-Tucker）方程解的方法所取代。

12.5.1　有约束最小化函数 fmincon

（1）功能：求多变量有约束非线性函数的最小值。

（2）调用格式如下：

- x = fmincon(fun,x0,A,b)。
- x = fmincon(fun,x0,A,b,Aeq,beq)。
- x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)。
- x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)。
- x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)。
- x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2, ...)。
- [x,fval] = fmincon(...)。
- [x,fval,exitflag] = fmincon(...)。

- [x,fval,exitflag,output] = fmincon(...)。
- [x,fval,exitflag,output,lambda] = fmincon(...)。
- [x,fval,exitflag,output,lambda,grad] = fmincon(...)。
- [x,fval,exitflag,output,lambda,grad,hessian] = fmincon(...)。

（3）描述如下：

- fmincon 用于求多变量有约束非线性函数的最小值。该函数常用于有约束非线性优化问题。
- x = fmincon(fun,x0,A,b)：给定初值 *x*0，求解 fun 函数的最小值 ***x***。fun 函数的约束条件为 *Ax*≤b，*x*0 可以是标量、向量或矩阵。
- x = fmincon(fun,x0,A,b,Aeq,beq)：最小化 fun 函数，约束条件为 Aeq*x* = beq 和 *Ax*≤*b*。若没有不等式存在，则设置 *A*=[]、*b*=[]。
- x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)：定义设计变量 ***x*** 的下界 lb 和上界 ub，使得总是有 lb≤*x*≤ub。若无等式存在，则令 Aeq=[]、beq=[]。
- x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)：在上面的基础上，在 nonlcon 参数中提供非线性不等式 *c*(*x*) ≤0 或等式 ceq(x)=0。fmincon 函数要求 *c*(*x*)≤0 且 ceq(*x*) = 0。当无边界存在时，令 lb=[]和（或）ub=[]。
- x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)：用 optiions 参数指定的参数进行最小化。
- x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2,...)：将问题参数 *P*1、*P*2 等直接传递给函数 fun 和 nonlin。若不需要这些变量，则传递空矩阵到 *A*、*b*、Aeq、beq、lb、ub、nonlcon 和 options。
- [x,fval] = fmincon(...)：返回解 ***x*** 处的目标函数值。
- [x,fval,exitflag] = fmincon(...)：返回 exitflag 参数，描述函数计算的退出条件。
- [x,fval,exitflag,output] = fmincon(...)：返回包含优化信息的输出参数 output。
- [x,fval,exitflag,output,lambda] = fmincon(...)：返回解 ***x*** 处包含拉格朗日乘子的 lambda 参数。
- [x,fval,exitflag,output,lambda,grad] = fmincon(...)：返回解 ***x*** 处 fun 函数的梯度。
- [x,fval,exitflag,output,lambda,grad,hessian] = fmincon(...)：返回解 ***x*** 处 fun 函数的 Hessian 矩阵。

（4）变量。

nonlcon 参数计算非线性不等式约束 *c*(*x*)≤0 和非线性等式约束 ceq(*x*)=0。它是一个包含函数名的字符串。该函数可以是 M 文件、内部文件或 MEX 文件。它要求输入一个向量 ***x***，返回两个变量——解 ***x*** 处的非线性不等式向量 ***c*** 和非线性等式向量 ***ceq***。例如，若 nonlcon='mycon'，则 M 文件 mycon.m 具有的形式如下：

```
function [c,ceq] = mycon(x)
c = ...       %计算 x 处的非线性不等式
ceq = ...     %计算 x 处的非线性等式
```

若还计算了约束的梯度，即

```
options = optimset('GradConstr','on')
```

则 nonlcon 函数必须在第 3 个和第 4 个输出变量中返回 c(x)的梯度 GC 和 ceq(x)的梯度 GCeq。当被调用的 nonlcon 函数只需要两个输出变量（此时优化算法只需要 c 和 ceq 的值，而不需要 GC 和 Gceq 的值）时，可以通过查看 nargout 的值来避免计算 GC 和 GCeq 的值。

```
function [c,ceq,GC,GCeq] = mycon(x)
   c = ...              %解 x 处的非线性不等式
   ceq = ...            %解 x 处的非线性等式
   if nargout > 2       %被调用的 nonlcon 函数，要求有 4 个输出变量
   GC = ...             %不等式的梯度
   GCeq = ...           %等式的梯度
end
```

若 nonlcon 函数返回长度为 m 的向量 $\boldsymbol{c}$ 和长度为 n 的向量 $\boldsymbol{x}$，则 $c(x)$的梯度 GC 是一个 $n\times m$ 的矩阵，其中 GC(i,j)是 $c(j)$对 $x(i)$的偏导数。同样，若 $\boldsymbol{ceq}$ 是一个长度为 p 的向量，则 ceq(x)的梯度 Gceq 是一个 $n\times p$ 的矩阵，其中 GCeq(i,j)是 ceq(j)对 $x(i)$的偏导数。

其他参数意义同前。

大型优化问题。

① 使用大型算法，必须在 fun 函数中提供梯度信息（options.GradObj 设置为 on）。如果没有梯度信息，则给出警告信息。

fmincon 函数允许 $g(x)$为一个近似梯度，但使用真正的梯度将使优化过程更具稳健性。

② 当对矩阵的二阶导数（即 Hessian 矩阵）进行计算以后，用该函数求解大型问题将更有效。但不需要求得真正的 Hessian 矩阵。如果能提供 Hessian 矩阵的稀疏结构的信息（用 options 参数的 HessPattern 属性），则 fmincon 函数可以算得 Hessian 矩阵的稀疏有限差分近似。

③ 若 $x0$ 不是严格可行的，则 fmincon 函数选择一个新的严格可行初始点。

④ 若 x 的某些元素没有上界或下界，则 fmincon 函数更希望对应的元素设置为 Inf（对于上界）或-Inf（对于下界），而不希望强制性地给上界赋一个很大的正值或给下界赋一个很小的负值。

⑤ 线性约束最小化课题中需要注意的问题：

- Aeq 矩阵中若存在密集列或近密集列，会导致满秩并使计算费时。
- fmincon 函数剔除 Aeq 中线性相关的行。此过程需要进行反复的因式分解，因此，如果相关行很多的话，计算将是一件很费时的事情。
- 每一次迭代都要用下式进行稀疏最小二乘求解：

$$Aeq=AeqTR^T$$

其中 R^T 为前提条件的乔累斯基因子。

中型优化问题。

① 如果用 Aeq 和 Beq 清楚地提供等式约束，将比用 lb 和 ub 获得更好的数值解。

② 在二次子问题中，若有等式约束并且因等式（Dependent Equalities）被发现和剔除的话，将在过程标题中显示 dependent。只有在等式连续的情况下，因等式才会被剔除。若等

式系统不连续，则子问题将不可行并在过程标题中打印 infeasible 信息。

（5）算法。

① 大型优化算法。若提供了函数的梯度信息，并且只有上下界存在或只有线性等式约束存在时，则 fmincon 函数将默认选择大型算法。该算法基于内部映射牛顿法（Interior-reflective Newton Method）的子空间置信域法（Subspace Trust-region），每一次迭代都与用 PCG 法求解大型线性系统得到的近似解有关。

② 中型优化算法。fmincon 函数使用序列二次规划法（SQP）。该算法中，每一步迭代都求解二次规划子问题，并用 BFGS 法更新拉格朗日 Hessian 矩阵。

（6）诊断。

求大型优化问题的代码中不允许上限和下限相等，即不能有 lb（2）==ub（2），否则给出出错信息：

```
Equal upper and lower bounds not permitted in this large-scale
method.
Use equality constraints and the medium-scale method instead.
```

若只有等式约束，仍然可以使用大型算法。当既有等式约束又有边界约束时，使用中型算法。

（7）局限性。

目标函数和约束函数都必须是连续的，否则可能会给出局部最优解；当问题不可行时，fmincon 函数将试图使最大约束值最小化；目标函数和约束函数都必须是实数；对于大型优化问题，使用大型优化算法时，用户必须在 fun 函数中提供梯度（options 参数的 GradObj 属性必须设置为 on），并且只可以指定上界和下界约束，或者只有线性约束存在，Aeq 的行数不能多于列数；现在，如果在 fun 函数中提供了解析梯度，选项参数 DerivativeCheck 不能与大型方法一起用，以比较解析梯度和有限差分梯度，可以通过将 options 参数的 MaxIter 属性设置为 0 来用中型方法核对导数，然后用大型方法求解问题。

12.5.2 有约束最小化应用

例 12-12，找到使函数 $f(x)=-x_1x_2x_3$ 最小化的值，其中 $0\leqslant x_1+2x_2+2x_3\leqslant 72$。

代码设置如下：

```
function f = ex10_12(x)
f = -x(1) * x(2) * x(3);
```

在 MATLAB 命令行窗口中输入下列代码：

```
x0 = [10; 10; 10];    % 初值
A = [-1 -2 -2; ...
1  2  2];
b = [0;72];
[x,fval] = fmincon(@x10_12,x0,A,b)
```

结果如下：

```
x =
   24.0000
```

```
  19.0000
  19.0000
fval =
 -10.4560e+003
```

12.6 目标规划

前面介绍的最优化方法只有一个目标函数，是单目标最优化方法。但是，在许多实际工程问题中，往往希望多个指标都达到最优值，所以它有多个目标函数。这种问题称为多目标最优化问题。

12.6.1 目标规划函数 fgoalattain

（1）功能：求解多目标达到问题。

（2）调用格式如下：

- x = fgoalattain(fun,x0,goal,weight)。
- x = fgoalattain(fun,x0,goal,weight,A,b)。
- x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq)。
- x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub)。
- x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon)。
- x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,... lb,ub,nonlcon,options)。
- x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,... lb,ub,nonlcon,options,P1,P2,...)。
- [x,fval] = fgoalattain(...)。
- [x,fval,attainfactor] = fgoalattain(...)。
- [x,fval,attainfactor,exitflag] = fgoalattain(...)。
- [x,fval,attainfactor,exitflag,output] = fgoalattain(...)。
- [x,fval,attainfactor,exitflag,output,lambda] = fgoalattain(...)。

（3）描述如下：

Fgoalattain 函数用于求解多目标达到问题。

- x = fgoalattain(fun,x0,goal,weight)：试图通过变化 *x* 来使目标函数 fun 达到 goal 指定的目标。初值为 *x*0，weight 参数指定权重。
- x = fgoalattain(fun,x0,goal,weight,A,b)：求解目标达到问题，约束条件为线性不等式 $Ax \leqslant b$。
- x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq)：求解目标达到问题，除提供上面的线性不等式以外，还提供线性等式 Aeq*x* = beq。当没有不等式存在时，设置 *A*=[]、*b*=[]。
- x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub)：为设计变量 *x* 定义下界 lb 和上界 ub 集合，这样始终有 lb≤*x*≤ub。

- x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon)：将目标达到问题归结为 nonlcon 参数定义的非线性不等式 $c(x)\leqslant 0$ 或非线性等式 ceq(x)=0。fgoalattain 函数优化的约束条件为 $c(x)\leqslant 0$ 和 ceq(x) = 0。若不存在边界，则设置 lb=[]和（或）ub=[]。
- x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon,... options)：用 options 中设置的优化参数进行最小化。
- x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon,... options,P1,P2,...)：将问题参数 $P1$、$P2$ 等直接传递给函数 fun 和 nonlcon。如果不需要参数 ***A***, ***b***、Aeq、beq、lb、ub、nonlcon 和 options，则将它们设置为空矩阵。
- [x,fval] = fgoalattain(...)：返回解 ***x*** 处的目标函数值。
- [x,fval,attainfactor] = fgoalattain(...)：返回解 ***x*** 处的目标达到因子。
- [x,fval,attainfactor,exitflag] = fgoalattain(...)：返回 exitflag 参数，描述计算的退出条件。
- [x,fval,attainfactor,exitflag,output] = fgoalattain(...)：返回包含优化信息的输出参数 output。
- [x,fval,attainfactor,exitflag,output,lambda] = fgoalattain(...)：返回包含拉格朗日乘子的 lambda 参数。

（4）变量。

① goal 变量。

目标希望达到的向量值。向量的长度与 fun 函数返回的目标数 F 相等。fgoalattain 函数试图通过最小化向量 ***F*** 中的值来达到 goal 参数给定的目标。

② nonlcon 函数。

nonlcon 函数计算非线性不等式约束 $c(x)\leqslant 0$ 和非线性等式约束 ceq(x)=0。nonlcon 函数是一个包含函数名的字符串，该函数可以是 M 文件、内部函数或 MEX 文件。nonlcon 函数需要输入向量 x，返回两个变量——x 处的非线性不等式向量 ***c*** 和 ***x*** 处的非线性等式向量 ***ceq***。例如，若 nonlcon='mycon'，则 M 文件的形式如下：

```
function [c,ceq] = mycon(x)
c = ...       %计算 x 处的非线性不等式
ceq = ...     %计算 x 处的非线性等式
```

若约束函数的梯度可以计算，且 options.GradConstr 设为 on，即

```
options = optimset('GradConstr','on')
```

则函数 nonlcon 也必须在第 3 个和第 4 个输出变量中输出 $c(x)$的梯度 GC 和 ceq(x)的梯度 GCeq。可以通过核对 nargout 参数来避免计算 GC 和 GCeq。

```
function [c,ceq,GC,GCeq] = mycon(x)
c = ...            %x 处的非线性不等式
ceq = ...          %x 处的非线性等式
if nargout > 2     %被调用的 nonlcon 函数，有 4 个输出
  GC = ...         %不等式的梯度
  GCeq = ...       %等式的梯度
end
```

若 nonlcon 函数返回长度为 *m* 的向量 ***c*** 和长度为 *n* 的 ***x***，则 *c*(*x*)的梯度 GC 是一个 *n*×*m* 的矩阵，其中 GC(*i*,*j*)是 *c*(*j*)对 *x*(*i*)的偏导数。同样，若 ***ceq*** 是一个 *p* 元素的向量，则 ceq(*x*)的梯度 GCeq 是一个 *n*×*p* 的矩阵，其中 GCeq(*i*,*j*)是 ceq(*j*)对 *x*(*i*)的偏导数。

③ options 变量。

优化参数选项。可以用 optimset 函数设置或改变这些参数的值。

- DerivativeCheck：比较用户提供的导数（目标函数或约束函数的梯度）和有限差分导数。
- Diagnostics：打印将要最小化或求解的函数的诊断信息。
- DiffMaxChange：变量中有限差分梯度的最大变化。
- DiffMinChange：变量中有限差分梯度的最小变化。
- Display：显示水平。设置为 off 时不显示输出；设置为 iter 时显示每一次迭代的输出；设置为 final 时只显示最终结果。
- GoalExactAchieve：使得目标个数刚好达到，不多也不少。
- GradConstr：用户定义的约束函数的梯度。
- GradObj：用户定义的目标函数的梯度。使用大型方法时必须使用梯度，对于中型方法则是可选项。
- MaxFunEvals：函数评价的允许最大次数。
- MaxIter：函数迭代的允许最大次数。
- MeritFunction：如果设置为 multiobj，则使用目标达到或最大最小化目标函数的方法；若设置为 singleobj，则使用 fmincon 函数计算目标函数。
- TolCon：约束矛盾的终止容限。
- TolFun：函数值处的终止容限。
- TolX：*x* 处的终止容限。

④ weight 变量。

weight 变量为权重向量，可以控制低于或超过 fgoalattain 函数指定目标的相对程度。当 goal 的值都是非零值时，为了保证活动对象超过或低于的比例相当，将权重函数设置为 abs(goal)。

> **注意**
>
> 当目标值中的任意一个为零时，设置 weight=abs(goal)将导致目标约束看起来更像硬约束，而不像目标约束；当加权函数 weight 为正时，fgoalattain 函数试图使对象小于目标值。为了使目标函数大于目标值，将权重 weight 设置为负；为了使目标函数尽可能地接近目标值，使用 GoalsExactAchieve 参数，将 fun 函数返回的第一个元素作为目标。

⑤ attainfactor 变量。

attainfactor 变量是超过或低于目标的个数。若 attainfactor 为负，则目标已经溢出；若 attainfactor 为正，则目标个数还未达到。

其他参数意义同前。

（5）算法。

多目标优化同时涉及一系列对象。fgoalattain 函数求解该问题的基本算法是目标达到法。该算法为目标函数建立起目标值。多目标优化的具体算法已经在前面进行了详细的介绍。

fgoalattain 函数使用序列二次规划法（SQP），前面已经进行了较多的介绍。算法中对于一维搜索和 Hessian 矩阵进行了修改。当有一个目标函数不再发生改善时，一维搜索终止。修改的 Hessian 矩阵借助于本问题的结构，也被采用。attainfactor 参数包含解处的 γ 值。γ 取负值时表示目标溢出。

（6）局限性。

目标函数必须是连续的。fgoalattain 函数将只给出局部最优解。

12.6.2　目标规划应用

例 12-13，考虑下列线性微分方程组的求解问题。

$$x=(\boldsymbol{A}+\boldsymbol{BKC})x+\boldsymbol{B}u$$

$$y=\boldsymbol{C}x$$

$$\boldsymbol{A}=\begin{bmatrix}-0.5 & 0 & 0\\ 0 & -2 & 10\\ 0 & 1 & -2\end{bmatrix}\quad \boldsymbol{B}=\begin{bmatrix}1 & 0\\ -2 & 2\\ 0 & 1\end{bmatrix}\quad \boldsymbol{C}=\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}\quad \boldsymbol{K}=\begin{bmatrix}-1 & -1\\ -1 & -1\end{bmatrix}$$

代码设置如下：

```
function F = ex10_13(K,A,B,C)
F = sort(eig(A+B*K*C)); % Evaluate objectives
```

在 MATLAB 命令行窗口中输入下列代码：

```
A = [-0.5 0 0; 0 -2 10; 0 1 -2];
B = [1 0; -2 2; 0 1];
C = [1 0 0; 0 0 1];
K0 = [-1 -1; -1 -1];                        % Initialize controller matrix
goal = [-5 -3 -1];                          % Set goal values for the eigenvalues
weight = abs(goal);                         % Set weight for same percentage
lb = -4*ones(size(K0));                     % Set lower bounds on the controller
ub = 4*ones(size(K0));                      % Set upper bounds on the controller
options = optimset('Display','iter');  % Set display parameter
[K,fval,attainfactor] = fgoalattain(@(K) ex10_14(K,A,B,C),...
 K0,goal,weight,[],[],[],[],lb,ub,[],options)
```

结果如下：

```
             Attainment      Max    Line search    Directional
Iter F-count  factor   constraint  steplength   derivative  Procedure
   0     6           0       1.88521
   1    12        1.031       0.02998      1    0.745
   2    18       0.3525       0.06863      1    -0.613
   3    24       -0.1706      0.1071       1    -0.223   Hessian modified
   4   30        -0.2236      0.06654      1    -0.234   Hessian modified twice
   5    36       -0.3568      0.007894     1    -0.0812
   6    42   -0.3645      0.000145     1    -0.164   Hessian modified
   7    48       -0.3645      0            1    -0.00515  Hessian modified
   8    54   -0.3675      0.0001546    1    -0.00812  Hessian modified twice
   9    60   -0.3889      0.008328     1    -0.00751  Hessian modified
```

```
  10    66       -0.3862      0            1    0.00568
  11    72   -0.3863      10.752e-013  1    -0.998   Hessian modified twice
K =
  -4.0000   -0.2564
  -4.0000   -4.0000
fval =
  -6.9313
  -4.1588
  -1.4099
attainfactor =
  -0.3863
```

例 12-14，某化工厂拟生产两种新产品 A 和 B，其生产设备费用分别为 A：3 万元/吨；B：6 万元/吨。这两种产品均将造成环境污染，设由公害所造成的损失可折算为 A：2 万元/吨；B：1 万元/吨。由于条件限制，工厂生产产品 A 和产品 B 的最大生产能力分别为每月 5 吨和 6 吨，而市场需要这两种产品的总量每月不少于 7 吨。试问工厂如何安排生产计划，在满足市场需要的前提下，使设备投资和公害损失均达最小？该工厂决策认为，这两个目标中环境污染应优先考虑，设备投资的目标值为 16 万元，公害损失的目标为 14 万元。

设工厂每月生产产品 A 为 x_1 吨，B 为 x_2 吨，设备投资费为 $f_1(x)$，公害损失费为 $f_2(x)$，则这个问题可表达为多目标优化问题。

代码设置如下：

```
function f=ex10_14(x)
      f(1)=3*x(1)+6*x(2);
      f(2)=2*x(1)+x(2);
```

给定目标，权重按目标比例确定，给出初值：

```
goal=[16 14];
weight=[16 14];
x0=[2 5];
```

给出约束条件的系数：

```
A=[1 0;0 1;-1 -1];
b=[5 6 -7];
lb=zeros(2,1);
 [x,fval,attainfactor,exitflag]= ...
fgoalattain(@ex10_14,x0,goal,weight,A,b,[],[],lb,[])
```

计算结果如下：

```
x =
   5.0000   9.0000
fval =
  27.0000   19.0000
attainfactor =
   0.6875
exitflag =
    4
```

故工厂每月生产产品 A 为 5 吨，B 为 2 吨。设备投资费和公害损失费的目标值分别为 27 万元和 12 万元。达到因子为 0.6875，计算收敛。

例 12-15，某厂生产两种产品 A 和 B，已知生产 A 产品 100kg 需 6 个工时，生产 B 产品 100kg 需 8 个工时。假定每日可用的工时数为 40，且希望不雇临时工，也不加班生产。这两

种产品每 100kg 均可获利 200 元。此外，有个顾客要求每日供应 B 种产品 600kg。问应如何安排生产计划？

这里设生产 A、B 两种产品的数量分别为 x_1 和 x_2（均以 100kg 计），为了使生产计划比较合理，要求用人数尽量少，获利尽可能多，另外 B 种产品的产量尽量多。

根据题意，代码设置如下：

```
function f=ex10_15(x)
      f(1)=6*x(1)+8*x(2);
      f(2)=-200*x(1)-200*x(2);
      f(3)=-x(2);
```

给定目标，权重按目标比例确定，给出初值：

```
goal=[40 -800 -6];
weight=[40 -800 -6];
x0=[2 2];
```

给出约束条件的系数：

```
A=[6 8;0 -1];
b=[40 -6];
lb=zeros(2,1);
options=optimset('MaxFunEvals',5000);  % 设置函数评价的最大次数为 5000 次
[x,fval,attainfactor,exitflag] =…
=fgoalattain(@ex10_15,x0,goal,weight,A,b,[],[],lb,[],[],options)
```

计算结果如下：

```
x =
   9.0398   1.9375
fval =
  27.7386 -795.4533  -1.9375
attainfactor =
  -0.0682
exitflag =
    0
```

经过 5 000 次迭代以后，生产 A、B 两种产品的数量各为 2 010.98kg 和 1 910.75kg。

例 12-16，某工厂因生产需要欲采购一种原材料，市场上的这种原料有两个等级，甲级原料单价 3 元/千克，乙级原料单价 2 元/千克。要求所花总费用不超过 300 元，购得原料总量不少于 200 千克，其中甲级原料不少于 40 千克。问如何确定最好的采购方案？

这里设 x_1、x_2 分别为采购甲级和乙级原料的数量（千克），要求采购总费用尽量少，采购总重量尽量多，采购甲级原料尽量多。

根据题意，代码设置如下：

```
function f=ex10_16(x)
f(1)=3*x(1)+ 2*x(2);
      f(2)=-x(1)- x(2);
      f(3)=-x(1);
```

给定目标，权重按目标比例确定，给出初值：

```
goal=[300 -200 40];
weight=[300 -200 40];
x0=[55 55];
```

给出约束条件的系数：

```
 A=[2 1;-1 -1;-1 0];
 b=[300 -200 -40];
 [x,fval,attainfactor,exitflag]=fgoalattain(@ex10_16,x0,goal,…
weiht,A,b,[],[],lb,[],[])
```

输出计算结果：

```
x =
    40   160
fval =
   440  -200   -40
attainfactor =
    0.4667
exitflag =
     4
```

所以，最好的采购方案是采购甲级原料和乙级原料分别为 40 千克和 160 千克。此时采购总费用为 440 元，总重量为 200 千克，甲级原料总重量为 40 千克。

12.7 最大最小化

通常我们遇到的都是目标函数的最大化和最小化问题，但是在某些情况下，则要求最大值的最小化才有意义。例如城市规划中需要确定急救中心、消防中心的位置，可取的目标函数应该是到所有地点最大距离的最小值，而不是到所有目的地的距离和为最小。这是两种完全不同的准则，在控制理论、逼近论、决策论中也使用最大最小化原则。

MATLAB 优化工具箱中采用序列二次规划法求解最大最小化问题。

12.7.1 最大最小化函数 fminimax

（1）功能：求解最大最小化问题。

（2）调用格式如下：

- x = fminimax(fun,x0)。
- x = fminimax(fun,x0,A,b)。
- x = fminimax(fun,x0,A,b,Aeq,beq)。
- x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub)。
- x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)。
- x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)。
- x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2,...)。
- [x,fval] = fminimax(...)。
- [x,fval,maxfval] = fminimax(...)。
- [x,fval,maxfval,exitflag] = fminimax(...)。
- [x,fval,maxfval,exitflag,output] = fminimax(...)。

- [x,fval,maxfval,exitflag,output,lambda] = fminimax(...)。

（3）描述如下：

fminimax 函数用于使多目标函数中的最坏情况达到最小化。给定初值估计，该值必须服从一定的约束条件。

- x = fminimax(fun,x0)：初值为 *x*0，找到 fun 函数的最大最小化解 *x*。
- x = fminimax(fun,x0,A,b)：给定线性不等式 $Ax \leqslant b$，求解最大最小化问题。
- x = fminimax(fun,x,A,b,Aeq,beq)：给定线性等式 Aeq*x* = beq，求解最大最小化问题。如果没有不等式存在，则设置 *A*=[]、*b*=[]。
- x = fminimax(fun,x,A,b,Aeq,beq,lb,ub)：为设计变量定义一系列下限 lb 和上限 ub，使得总有 lb≤*x*≤ub。
- x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)：在 nonlcon 参数中给定非线性不等式约束 *c*(*x*)≤0 或等式约束 ceq(*x*)=0，fminimax 函数要求 *c*(*x*)≤0 且 ceq(*x*) = 0。若没有边界存在，则设置 lb=[]和（或）ub=[]。
- x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)：用 options 给定的参数进行优化。
- x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2,...)：将问题参数 *P*1、*P*2 等直接传递给函数 fun 和 nonlcon。如果不需要变量 *A*、*b*、Aeq、beq、lb、ub、nonlcon 和 options 则将它们设置为空矩阵。
- [x,fval] = fminimax(...)：返回解 *x* 处的目标函数值。
- [x,fval,maxfval] = fminimax(...)：返回解 *x* 处的最大函数值。
- [x,fval,maxfval,exitflag] = fminimax(...)：返回 exitflag 参数，描述函数计算的退出条件。
- [x,fval,maxfval,exitflag,output] = fminimax(...)：返回描述优化信息的结构输出 output 参数。
- [x,fval,maxfval,exitflag,output,lambda] = fminimax(...)：返回包含解 *x* 处拉格朗日乘子的 lambda 参数。

（4）变量：maxfval 变量。

maxfval 变量用于解 *x* 处函数值的最大值，即 maxfval = max{fun(*x*)}。

（5）算法。

fminimax 函数使用序列二次规划法（SQP）进行计算，对一维搜索法和 Hessian 矩阵的计算进行了修改。当有一个目标函数不再发生改善时，一维搜索终止。修改的 Hessian 矩阵借助于本问题的结构，也被采用。

（6）局限性。

目标函数必须连续，否则 fminimax 函数有可能给出局部最优解。

12.7.2　最大最小化应用

例 12-17，找到如下函数的最大最小值。

$$[f_1(x),f_2(x),f_3(x),f_4(x),f_5(x)]$$

其中，

$$f_1(x)=2x_1^2+x_2^2-48x_1-40x_2+304$$

$$f_2(x)=-x_1^2-3x_2^2$$

$$f_3(x)=x_1+3x_2-18$$

$$f_4(x)=-x_1-x_2$$

$$f_5(x)=x_1+x_2-8$$

代码设置如下：

```
function f = ex10_17(x)
f(1)= 2*x(1)^2+x(2)^2-48*x(1)-40*x(2)+304;    % Objectives
f(2)= -x(1)^2 - 3*x(2)^2;
f(3)= x(1) + 3*x(2) -18;
f(4)= -x(1)- x(2);
f(5)= x(1) + x(2) - 8;
```

在 MATLAB 命令行窗口中输入下列代码：

```
 x0 = [0.1; 0.1];   % 提供解的初值
 [x,fval] = fminimax(@ex10_17,x0)
```

经过 7 次迭代以后，结果如下：

```
x =
   4.0000
   4.0000
fval =
   0.0000  -64.0000   -9.0000   -8.0000   -0.0000
```

例 12-18，定位问题。设某城市有某种物品的 10 个需求点，第 i 个需求点 P_i 的坐标为 (a_i,b_i)，道路网与坐标轴平行，彼此正交。现打算建一个该物品的供应中心，且由于受到城市某些条件的限制，该供应中心只能设在 x 介于[5,8]、y 介于[5,8]的范围内。问该中心应建在何处为好？

P_i 点的坐标为：

a_i:　2　1　5　9　3　12　6　20　18　11

b_i:　10 9　13 18　1　3　5　7　8　6

设供应中心的位置为 (x,y)，要求它到最远需求点的距离尽可能小。由于此处应采用沿道路行走的距离，可知用户 P_i 到该中心的距离为 $|x-a_i|+|y-b_i|$。

根据题意，代码设置如下，并保存在 ex10_18.m 文件中：

```
function f = ex10_18(x)
%输入各个点的坐标值
a=[2  1  5  9  3  12  6  20  18  11];
b=[10 9 13 18  1  3   5  7   8   6];
f(1) = abs(x(1)-a(1))+abs(x(2)-b(1));
f(2) = abs(x(1)-a(2))+abs(x(2)-b(2));
```

```
    f(3) = abs(x(1)-a(3))+abs(x(2)-b(3));
    f(4) = abs(x(1)-a(4))+abs(x(2)-b(4));
    f(5) = abs(x(1)-a(5))+abs(x(2)-b(5));
    f(6) = abs(x(1)-a(6))+abs(x(2)-b(6));
    f(7) = abs(x(1)-a(7))+abs(x(2)-b(7));
    f(8) = abs(x(1)-a(8))+abs(x(2)-b(8));
    f(9) = abs(x(1)-a(9))+abs(x(2)-b(9));
    f(10) = abs(x(1)-a(10))+abs(x(2)-b(10));
```

在 MATLAB 命令行窗口中输入下列代码：

```
 x0 = [6; 6];      % 提供解的初值
AA=[-1 0
1  0
  0 -1
 0  1];
>>bb=[-5;8;-5;8];
>> [x,fval] = fminimax(@ex10_18,x0,AA,bb)
```

计算结果如下：

```
x =
    8.0000
    7.0000
fval =
    9.0000    9.0000    9.0000   19.0000   11.0000    8.0000    4.0000   19.0000
   11.0000    4.0000
```

可见，在限制区域内的东北角设置供应中心，可以使该点到各需求点的最大距离最小。最大最小距离为 14 个距离单位。

12.8　本章小结

本章主要介绍了 MATLAB 中的优化工具箱，包括优化工具箱中的函数、最小化问题、线性规划问题、无约束非线性规划问题、二次规划问题、有约束最小化问题、目标规划问题及最大最小化问题等。这些最优化问题在 MATLAB 中都可以找到相应的函数，用户可以参考每节中的函数功能描述及相应示例。

在解决优化问题时，首先，要根据实际情况抽象出一个数学模型；其次，要弄清楚模型是否有限制条件或约束，同时还要搞清楚目标函数是否是线性的；最后，利用相应优化函数有针对性地解决实际问题。最优化方法的发展很快，现在已经包含多个分支，如线性规划、整数规划、非线性规划、动态规划、多目标规划等。由于篇幅所限，这里仅仅讨论了一些常见的规划问题。

第 13 章

小波分析工具箱

知识要点

MATLAB 小波分析工具箱提供了一个可视化的小波分析工具，是一个很好的算法研究、工程设计、仿真和应用平台，特别适合于信号和图像分析、综合、去噪、压缩等领域的研究人员。本章的主要结构：第 1 节介绍小波分析的基本数学理论及小波应用的演示示例，第 2 节介绍小波工具箱的使用，第 3 节介绍小波分析在图像处理中的具体应用，让读者亲身感受 MATLAB 工具箱的强大魅力。

学习要求	知识点 \ 学习目标	了解	理解	应用	实践
	小波分析的发展历程	√			
	小波分析理论	√			
	小波工具箱			√	√
	一维连续（离散）小波工具箱			√	√
	小波分析在图像处理中的应用		√		√
	小波工具箱中的基本函数		√		

13.1 小波分析

小波分析是近年来发展起来的一种新的时频分析方法，其主要应用于信号处理方面。小波分析在图像处理中也有非常重要的应用，包括图像压缩、图像去噪、图像融合、图像分解、图像增强等。下面介绍小波的基本知识。

13.1.1 小波分析概述

小波变换是一种新的变换分析方法，它的主要特点是通过变换能够充分突出问题某些方面的特征。因此，小波变换在许多领域得到了成功的应用，特别是小波变换的离散数字算法已被广泛用于许多问题的变换研究中。

从小波变换的发展过程来说，大致可分成 3 个阶段。

第 1 阶段：孤立应用时期。主要特征是一些特殊构造的小波在某些科学研究领域的特定问题上的应用。这个时期的代表性工作是法国地球物理学家 Morlet 和 Grossmann 第一次把“小波”用于分析处理地质数据，引进了以他们的名字命名的时间-尺度小波，即 Grossmann-Morlet 小波。

同时，著名的计算机视觉专家 Marr 在他的“零交叉”理论中使用的可按“尺度大小”变化的滤波算子，现在称为“墨西哥帽”的小波也是这个时期有名的工作成果之一，这部分工作成果和后来成为 Mallat 的正交小波构造理论支柱的“多尺度分析”或“多分辨分析”有密切联系。这个时期一个有趣的现象是各个领域的专家、学者和工程师在完全不了解别人的研究工作的状态下巧妙、独立地构造自己需要的“小波”。虽然如此，但通观全局可以发现，这些专家、学者和工程师所从事研究的领域广泛分布于科学和技术研究的许多方面，因此，这个现象从另一个侧面预示着小波分析理论研究和应用热潮的到来，说明了小波理论产生的历史必然性。

第 2 阶段：国际性研究热潮和统一构造时期。真正的小波热潮开始于 1986 年，当时法国数学家 Meyer 成功地构造出具有一定衰减性质的光滑函数，这个函数（算子）的二进尺度伸缩和二进整倍数平移产生的函数系构成著名的 2-范数函数空间的标准正交基。进入这个时期之后，Lemarie 和 Battle 又分别独立地构造得到了这样“好的”小波。后来 Meyer 和计算机科学家 Mallat 提出多分辨分析概念，成功地统一了此前 Strömberg、Meyer、Lemarie 和 Battle 的不同的小波构造方法。同时，Mallat 还在多分辨分析的基础上简洁地得到了离散小波的数值算法，即现在的 Mallat 分解和合成算法，并且将此算法用于数字图像的分解与重构。几乎同时，比利时数学家 Daubechies 基于多项式方式构造出具有有限支撑的正交小波基和对称双正交小波；Chui 和中国学者王建忠基于样条函数构造出单正交小波函数，并讨论了具有最好局部化性质的尺度函数和小波函数的构造方法。

第 3 阶段：全面应用时期。从 1992 年开始，小波分析方法进入全面应用阶段。在前一阶段研究工作的基础上，特别是数字信号和数字图像的 Mallat 分解和重构算法的确定，使小

波分析的应用迅速波及科学研究和工程技术应用研究的几乎所有领域。

鉴于小波分析的“自适应性质”和“数学显微镜性质”，使其广泛应用于基础科学、应用科学，尤其是信息科学、信号分析的方方面面，如图像处理与传输、信号处理、模式识别、地震探测、音乐、雷达、CT 成像、彩色复印等。

由于小波分析在数字信号分析方面独特的诊断效果，来自不同学科、不同背景、不同兴趣爱好的科技工作者自觉投入到小波分析理论与应用研究中，涌现出一批高水平的论文和著作，在国内外形成一次又一次研究高潮，至今方兴未艾。

从小波变换的数学理论来说，它是继 Fourier 变换之后纯粹数学和应用数学完美结合的又一光辉典范，享有“数学显微镜”的美称。从纯粹数学的角度来说，研究的小波变换是调和分析（其中包括函数空间、广义函数、Fourier 分析、抽象调和分析等）这一重要学科大半个世纪以来的工作结晶；从应用科学和技术科学的角度来说，小波变换又是计算机应用、信号处理、图像分析、非线性科学和工程技术近年来在方法上的重大突破。

实际上，由于小波变换在它的产生、发展、完善和应用的整个过程中都广泛受惠于计算机科学、信号和图像处理科学、应用数学和纯粹数学、物理科学和地球科学等众多科学研究领域和工程技术应用领域的专家、学者、工程师们的共同努力，所以，现在它已成为科学研究和工程技术应用中涉及面极广泛的一个热门话题。

13.1.2　傅里叶变换和小波变换

在传统的傅里叶分析中，信号完全是在频域展开的，不包含任何时频的信息，这对于某些应用来说是很恰当的，因为信号频率的信息对其是非常重要的。但其丢弃的时域信息可能对某些应用同样非常重要，所以人们对傅里叶分析进行了推广，提出了很多能表征时域和频域信息的信号分析方法，如短时傅里叶变换、Gabor 变换、时频分析、小波变换等。

其中短时傅里叶变换是在傅里叶分析基础上引入时域信息的最初尝试，其基本假定在于在一定的时间窗内信号是平稳的，那么通过分割时间窗，在每个时间窗内把信号展开到频域就可以获得局部的频域信息，但是它的时域区分度只能依赖于大小不变的时间窗，对某些瞬态信号来说还是粒度太大。换言之，短时傅里叶分析只能在一个分辨率上进行，所以对很多应用来说不够精确，存在很大的缺陷。

而小波分析则克服了短时傅里叶变换在单分辨率上的缺陷，具有多分辨率分析的特点，在时域和频域都有表征信号局部信息的能力，时间窗和频率窗都可以根据信号的具体形态动态调整，一般情况下，在低频部分（信号较平稳）可以采用较低的时间分辨率提高频率的分辨率，在高频部分（频率变化不大）可以用较低的频率分辨率来换取精确的时间定位。因为这些特征，小波分析可以探测正常信号中的瞬态，并展示其频率成分，被称为数学显微镜，广泛应用于各个时频分析领域。

小波分析是傅里叶分析思想方法的发展与延拓。它自产生以来，就一直与傅里叶分析密切相关。它的存在证明小波基的构造及结果分析都依赖于傅里叶分析，二者是相辅相成的。两者相比较主要有以下不同。

（1）傅里叶变换的实质是把能量有限信号 $f(t)$分解到以{ $e^{j\omega t}$ }为正交基的空间上去；小波变换的实质是把能量有限信号 $f(t)$ 分解到W_{-j} （j=1,2,⋯,J）和V_{-j} 所构成的空间上去。

（2）傅里叶变换用到的基本函数只有$\sin(\omega t)$、$\cos(\omega t)$、$\exp(i\omega t)$，具有唯一性；小波分析用到的函数（即小波函数）则不具有唯一性，同一个工程问题用不同的小波函数进行分析有时结果相差甚远。小波函数的选用是小波分析应用到实际中的一个难点问题（也是小波分析研究的一个热点问题），目前往往是通过经验或不断地试验（与结果进行对照分析）来选择小波函数。

（3）在频域中，傅里叶变换具有较好的局部化能力，特别是对于那些频率成分比较简单的确定性信号，傅里叶变换很容易把信号表示成各频率成分的叠加和的形式。例如，$\sin(\omega_1 t)+0.345\sin(\omega_2 t)+4.23\cos(\omega_3 t)$。但在时域中，傅里叶变换没有局部化能力，即无法从信号 $f(t)$ 的傅里叶变换 $\hat{f}(\omega)$ 中看出 $f(t)$ 在任一时间点附近的性态。

（4）在小波分析中，尺度 a 的值越大，相对于傅里叶变换中ω 的值越小。

（5）在短时傅里叶变换中，变换系数$S(\omega,\tau)$主要依赖于信号在$[\tau-\delta,\tau+\delta]$片段中的情况，时间宽度是$2\delta$（因为$\delta$ 是由窗函数$g(t)$唯一确定的，所以2δ 是一个定值）。在小波变换中，变换系数$W_f(a,b)$主要依赖于信号在$[b-a\Delta\psi,b+a\Delta\psi]$片段中的情况，时间宽度是$2a\Delta\psi$，该时间宽度是随着尺度 a 变化而变化的，所以小波变换具有时间局部分析能力。

（6）若用信号通过滤波器来解释，小波变换与短时傅里叶变换的不同之处在于：对短时傅里叶变换来说，带通滤波器的带宽Δf 与中心频率 f 无关；相反，小波变换带通滤波器的带宽Δf 则正比于中心频率 f，即

$$Q=\frac{\Delta f}{f}=\mathrm{C}$$

其中，C 为常数，表示滤波器有一个恒定的相对带宽，称为等 Q 结构（Q 为滤波器的品质因数，且$Q=\dfrac{中心频率}{带宽}$）。

1．傅里叶变换

在信号处理中的重要方法之一是傅里叶变换，它架起了时域和频域之间的桥梁。

对很多信号来说，傅里叶分析非常有用，因为它能给出信号令包含的各种频率成分。但是，傅里叶变换有着严重的缺点：变换之后使信号失去了时间信息，它不能告诉人们在某段时间里发生了什么变化。而很多信号都包含人们感兴趣的非稳态（或者瞬变）特性，如漂移、趋势项、突然变化及信号的开始或结束。这些特性是信号的最重要部分。因此傅里叶变换不适于分析处理这类信号。

虽然傅里叶变换能够将信号的时域特征和频域特征联系起来，能分别从信号的时域和频域观察，但却不能把二者有机地结合起来。这是因为信号的时域波形中不包含任何频域信息。而傅里叶谱是信号的统计特性，从其表达式中也可以看出，它是整个时间域内的积分，没有

局部化分析信号的功能，完全不具备时域信息。也就是说，对于傅里叶谱中的某一频率，不知道这个频率是在什么时候产生的。这样在信号分析中就面临一对最基本的矛盾：时域和频域的局部化矛盾。

在实际的信号处理过程中，尤其是对非平稳信号的处理中，信号在任一时刻附近的频域特征都很重要。如柴油机缸盖表面的震动信号就是由撞击或冲击产生的，是一个瞬变信号，仅从时域或频域上来分析是不够的。这就促使人们去寻找一种新方法，能够将时域和频域结合起来描述观察信号的时频联合特征，构成信号的时频谱。这就是所谓的时频分析法，也称为时频局部化方法。

2．小波变换

（1）连续小波变换。

如果函数$\psi(x)$满足以下容许性条件：

$$C_\psi=\int\frac{\left|\hat{\psi}(\omega)\right|^2}{|\omega|}\mathrm{d}\omega<\infty$$

则称$\psi(x)$为一个基本小波，并定义如下的积分变换：

$$\left(W_\psi f\right)(a,b)=\left|a\right|^{-\frac{1}{2}}\int f(x)\overline{\psi\left(\frac{x-b}{a}\right)}\mathrm{d}x，\quad f(x)\in L^2\left(R\right)$$

上述积分变换为$f(x)$以$\psi(x)$为基的积分连续小波变换，a为尺度因子，表示与频率相关的伸缩，b为时间平移因子。

如果$\hat{\psi}(\omega)$是连续的，则容易得到：

$$\hat{\psi}(0)=0\Leftrightarrow\int_{-\infty}^{+\infty}\psi(t)\mathrm{d}t=0$$

$\psi(t)$又称为母小波，因为其伸缩、平移可构成$L^2(R)$的一个标准正交基。

很显然，并非所有函数都能保证上式中表示的变换对于所有$f\in L^2(R)$均有意义。另外，在实际应用尤其是信号处理及图像处理的应用中，变换只是一种简化问题、处理问题的有效手段，最终目的需要回到原问题的求解，因此，还要保证连续小波变换存在逆变换。同时，作为窗口函数，为了保证时间窗口与频率窗口具有快速衰减特性，经常要求函数$\psi(x)$具有如下性质：

$$|\psi(x)|\leqslant C(1+|x|)^{-1-\varepsilon}，\quad|\hat{\psi}(v)|\leqslant C(1+|\omega|)^{-1-\varepsilon}$$

其中，C 为与x、v无关的常数，$\varepsilon>0$。

（2）离散小波变换。

$(w_\psi f)(a,b)=<f(t),\psi_{a,b}(t)>$，将$a$、$b$离散化，令$a=2^{-j}$，$b=2^{-j}k$，$j,k\in Z$，可得离散小波变换：

$$(DW_\psi f)(j,k)=<f(t),\psi_{j,k}(t)>$$

其中，$\psi_{j,k}(t)=2^{\frac{j}{2}}\psi(2^jt-k)$，$j,k\in Z$。

由此可见，一维信号 $f(x)$ 经离散小波变换得到的是一个二维数组。

13.1.3　多分辨分析

小波理论包括连续小波和二进小波变换，在映射到计算域的时候存在很多问题，因为两者都存在信息冗余，在对信号采样以后，需要计算的信息量还是相当大的，尤其是连续小波变换，因为要对精度内所有的尺度和位移都进行计算，所以计算量相当大。

而二进小波变换虽然在离散的尺度上进行伸缩和平移，但是小波之间没有正交性，各个分量的信息掺杂在一起，为分析带来了不便。

真正使小波在应用领域得到较大发展的是 Meyer 在 1986 年提出的一组小波，其二进制伸缩和平移构成 $L^2(R)$ 的标准化正交基。

在此结果基础上，1988 年 S.Mallat 在构造正交小波时提出了多分辨分析的概念，从函数分析的角度给出了正交小波的数学解释，在空间的概念上形象地说明了小波的多分辨率特性，给出了通用的构造正交小波的方法，并将之前所有的正交小波构造方法统一起来，在傅里叶分析中的快速傅里叶算法基础上，给出了小波变换的快速算法——Mallat 算法。这样，在计算上变得可行以后，小波变换在各个领域才发挥它独特的优势，解决了各类问题，为人们提供了更多的关于时域分析的信息。

形象一点说，多分辨分析就是要构造一组函数空间，每组空间的构成都有一个统一的形式，而所有空间的闭包则逼近 $L^2(R)$。在每个空间中，所有的函数都构成该空间的标准化正交基，而所有函数空间的闭包中的函数则构成 $L^2(R)$ 的标准化正交基。那么，如果对信号在这类空间上进行分解，就可以得到相互正交的时频特性。而且由于空间数目是无限可数的，可以很方便地分析所关心的信号的某些特性。

下面简要介绍一下多分辨分析的数学理论。

设 $\{V_j\}_{j\in Z}$ 是 $L^2(R)$ 上的一列闭子空间，$\varphi(t)$ 是 $L^2(R)$ 中的一个函数，如果它们满足如下的 5 个条件，即

① 调性：对任何整数 $j\in Z$，$V_j \subset V_{j+1}$；

② 唯一性：$\bigcap_{j\in Z} V_j = \{0\}$；

③ 密性：$\overline{\bigcup_{j\in Z} V_j} = L^2(R)$；

④ 缩性：$f(t)\in V_j \Leftrightarrow f(2t)\in V_{j+1}$，$\forall j\in Z$；

⑤ 正交基的存在性：存在 $\{\varphi(t-k)\}_{k\in Z}$ 构成子空间 V_0 的标准正交基。

那么称 $\varphi(t)$ 是 $L^2(R)$ 正交多分辨分析的窗口函数。

多分辨空间的关系可用图 13-1 形象地说明。

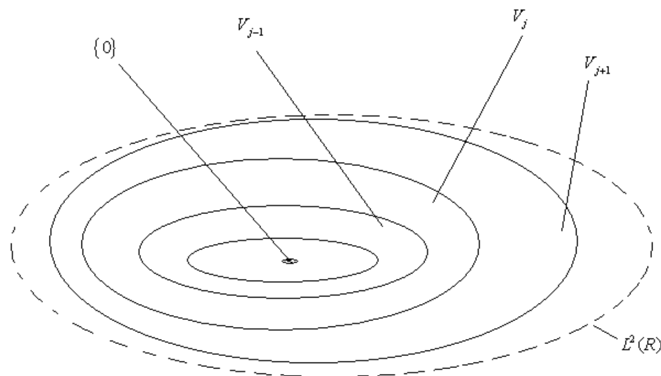


图 13-1　多分辨空间关系

由多分辨分析的定义容易得到一个重要结果，即函数族$\{\phi_{j,k}(t)\}_{k\in Z}$是$V_j$空间的标准正交基。下面将要讨论的是如何由这个正交多分辨分析去构造$L^2(R)$上的一个正交小波$\psi(t)$，使$\{\psi_{j,k}(t)\}_{j,k\in Z}$构成$L^2(R)$的标准正交小波基。

仿照 Shannon 小波的构造方法，对$\forall j\in Z$，定义如下的子空间W_j：

$$W_j \perp V_j，\ V_{j+1}=W_j\oplus V_j$$

容易验证，子空间序列$\{W_j\}_{j\in Z}$具有下述性质：

①$\forall j\neq l$，$W_j\perp W_l$；

②$L^2(R)=\bigoplus_{l=-\infty}^{+\infty}W_l$；

③$\forall j\in Z$，$g(t)\in W_j \Leftrightarrow g(2t)\in W_{j+1}$。

因此，根据②可知，为了得到空间$L^2(R)$的标准正交基，只需构造每一个子空间W_j的标准正交基；再由③得到，这只需构造W_0的标准正交基就足够了。

下面介绍塔式算法。

（1）信号在小波空间的展开为：

$$f(t)=\sum_{j\in Z}f_{W_j}=\sum_{j\in Z}\sum_{k\in Z}<f(t),\psi_{j,k}(t)>\psi_{j,k}(t)$$

（2）小波分解算法。

使用多分辨分析的金字塔算法：

$$f(t)=\sum_{k\in Z}<f(t),\varphi_{j,k}(t)>\varphi_{j,k}(t)+\sum_{j\le j'<J}\sum_{k\in Z}<f(t),\psi_{j'k}(t)>\psi_{j'k}(t)$$

实际计算时，可以一次一次地进行小波分解，然后递推实现（J-j）次小波分解。不妨记一次小波分解的尺度系数和小波系数为：

$$\begin{cases} c_{j,k}=<f,\varphi_{j,k}> \\ d_{j,k}=<f,\psi_{j,k}> \end{cases}$$

可以知道：

$$\varphi_{j,k}=\sum_{n}<\varphi_{j,k},\varphi_{j+1,n}>\varphi_{j+1,n}$$

$$<\varphi_{j,k},\varphi_{j+1,n}>=\frac{2^j}{\sqrt{2}}\int_{-\infty}^{+\infty}\varphi(2^j t-k)\overline{\varphi(2^{j+1}t-n)}\mathrm{d}t$$

$$=\frac{1}{\sqrt{2}}\int_{-\infty}^{+\infty}\varphi(t')\overline{\varphi(2t'+2k-n)}\mathrm{d}t$$

因为

$$h_k=<\varphi(t),\varphi(2t-k)>=\int_{-\infty}^{+\infty}\varphi(t)\overline{\varphi(2t-k)}\mathrm{d}t$$

故

$$<\varphi_{j,k}(t),\varphi_{j+1,n}(t)>=\frac{1}{\sqrt{2}}h_{n-2k}$$

代入得

$$\varphi_{j,k}=\frac{1}{\sqrt{2}}\sum_{n}h_{n-2k}\varphi_{j+1,n}=\frac{1}{\sqrt{2}}\sum_{n'}h_{n'}\varphi_{j+1,n'+2k}$$

从而

$$c_{j,k}=<f(t),\varphi_{j,k}>=\frac{1}{\sqrt{2}}\sum_{n}h_n<f(t),\varphi_{j+1,n+2k}>$$

$$=\frac{1}{\sqrt{2}}\sum_{n}h_n c_{j+1,n+2k}$$

得到如下递推公式

$$c_{j,k}=\frac{1}{\sqrt{2}}\sum_{n\in Z}h_n c_{j+1,n+2k}$$

相应得到

$$d_{j,k}=\frac{1}{\sqrt{2}}\sum_{n\in Z}g_n c_{j+1,n+2k}$$

可以很快计算出尺度系数和小波系数，这就是著名的 Mallat 分解算法。

> ○ 注意
>
> 如果需要，可以对信号进行多级小波分解，例如对上述分解结果中的近似序列再做一次分解，可得到下一级分解系数，以此类推，直到任何指定级数的多级小波分解为止。最后需要保留的数据是最低一级近似序列、最低一级和所有高级细节序列。

其他各级近似序列无须保留，因为采用重构算法，可以利用某一级近似序列和同级细节序列重构上一级近似序列。又注意到，由于本算法的采样过程中，每经过一次分解，近似序列被分解为低一级的近似序列和细节序列，但两者的长度均为输入序列的一半，即数据总量保持不变。相应的重构算法可详见参考文献，这里就不做详细讨论了。

13.1.4　小波包分析

短时傅里叶变换对信号的频带划分是线性等间隔的。多分辨分析可以对信号进行有效的时频分解，但由于其尺度是按二进制变化的，所以在高频频段其频率分辨率较差，而在低频频段其时间分辨率较差，即对信号的频带进行指数等间隔划分（具有等 Q 结构）。

小波包分析能够为信号提供一种更精细的分析方法，它将频带进行多层次划分，对多分辨率分析没有细分的高频部分进一步分解，并能够根据被分析信号的特征，自适应地选择相应频带，使之与信号频谱相匹配，从而提高时频分辨率，因此小波包具有更广泛的应用价值。

关于小波包分析的理解，这里以一个 3 层的分解进行说明，其小波包分解树如图 13-2 所示。

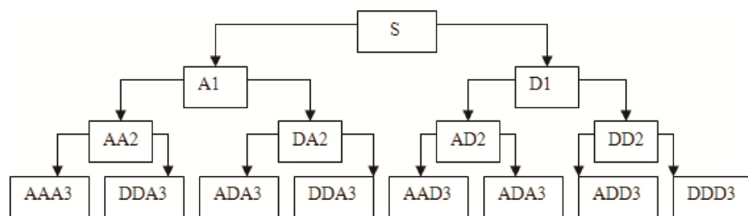


图 13-2　小波包分解树

图 13-2 中，A 表示低频，D 表示高频，末尾的序号数表示小波分解的层数（即尺度数）。分解具有关系：S=AAA3+DDA3+ADA3+DDA3+AAD3+ADA3+ADD3+DDD3。

1. 小波包的定义

在多分辨分析中，$L^2(R)=\bigoplus_{j\in Z} W_j$，表明多分辨分析是按照不同的尺度因子 j 把 Hilbert 空间 $L^2(R)$ 分解为所有子空间 $W_j(j\in Z)$ 的正交和的。其中，W_j 为小波函数 $\Psi(t)$ 的闭包（小波子空间）。现在，希望对小波子空间 W_j 按照二进制分式进行频率的细分，以达到提高频率分辨率的目的。

一种自然的做法是将尺度空间 V_j 和小波子空间 W_j 用一个新的子空间 U_j^n 统一起来表

征，若令

$$\begin{cases} U_j^0 = V_j \\ U_j^1 = W_j \end{cases} \quad j \in Z$$

则 Hilbert 空间的正交分解$V_{j+1} = V_j \oplus W_j$，即可用U_j^n的分解统一为

$$U_{j+1}^0 = U_j^0 \oplus U_j^1 \quad j \in Z$$

定义子空间U_j^n是函数$U_n(t)$的闭包空间，而$U_n(t)$是函数$U_{2n}(t)$的闭包空间，并令$U_n(t)$满足下面的双尺度方程：

$$\begin{cases} u_{2n}(t) = \sqrt{2}\sum_{k \in Z} h(k)u_n(2t-k) \\ u_{2n+1}(t) = \sqrt{2}\sum_{k \in Z} g(k)u_n(2t-k) \end{cases}$$

式中，$g(k) = (-1)^k h(1-k)$，即两系数也具有正交关系。当 n=0 时，以上两式直接给出

$$\begin{cases} u_0(t) = \sum_{k \in Z} h_k u_0(2t-k) \\ u_1(t) = \sum_{k \in Z} g_k u_0(2t-k) \end{cases}$$

在多分辨分析中，$\phi(t)$和$\psi(t)$满足双尺度方程：

$$\begin{cases} \phi(t) = \sum_{k \in Z} h_k \phi(2t-k) & \{h_k\}_{k \in Z} \in l^2 \\ \psi(t) = \sum_{k \in Z} g_k \phi(2t-k) & \{g_k\}_{k \in Z} \in l^2 \end{cases}$$

相比较，$u_0(t)$和$u_1(t)$分别退化为尺度函数$\phi(t)$和小波基函数$\psi(t)$。这样上式就是相互等价的。把这种等价表示推广到$n \in Z_+$（非负整数）的情况，即得到

$$U_{j+1}^n = U_j^n \oplus U_j^{2n+1} \quad j \in Z;\quad n \in Z_+$$

定义（小波包）序列$\{u_n(t)\}$（其中$n \in Z_+$）称为由基函数$u_0(t) = \phi(t)$确定的正交小波包。

由于$\phi(t)$由h_k唯一确定，所以又称$\{u_n(t)\}_{n \in Z}$为关于序列$\{h_k\}$的正交小波包。

2．小波包的性质

定理 1：设非负整数 n 的二进制表示为 $n=\sum_{i=1}^{\infty}\varepsilon_i 2^{i-1}$，$\varepsilon_i=0$ 或 1，则小波包 $\hat{u}_n(\omega)$ 的傅里叶变换由下式给出：

$$\hat{u}_n(\omega)=\prod_{i=1}^{\infty} m_{\varepsilon_i}(\omega/2^{j})$$

式中

$$m_0(\omega)=H(\omega)=\frac{1}{\sqrt{2}}\sum_{k=-\infty}^{+\infty} h(k)\mathrm{e}^{-jk\omega}$$

$$m_1(\omega)=G(\omega)=\frac{1}{\sqrt{2}}\sum_{k=-\infty}^{\infty} g(k)\mathrm{e}^{-jk\omega}$$

定理 2：设 $\{u_n(t)\}_{n\in Z}$ 是正交尺度函数 $\varphi(t)$ 的正交小波包，则 $<u_n(t-k),u_n(t-l)>=\delta_{kl}$，即 $\{u_n(t)\}_{n\in Z}$ 构成 $L^2(R)$ 的规范正交基。

3．小波包的空间分解

令 $\{u_n(t)\}_{n\in Z}$ 是关于 h_k 的小波包族，考虑用下列方式生成子空间族。现在令 n=1,2,⋯，j=1,2,⋯，并对小波子空间 W_j 式作迭代分解，则有

$$W_j=U_j^1=U_{j-1}^2\oplus U_{j-1}^3$$
$$U_{j-1}^2=U_{j-2}^4\oplus U_{j-2}^5,\quad U_{j-2}^5=U_{j-2}^6\oplus U_{j-2}^7$$

因此，很容易得到小波子空间 W_j 的各种分解：

$$W_j=U_{j-1}^2\oplus U_{j-1}^3$$
$$W_j=U_{j-2}^4\oplus U_{j-2}^5\oplus U_{j-2}^6\oplus U_{j-2}^7$$
$$\cdots$$
$$W_j=U_{j-k}^{2^k}\oplus U_{j-k}^{2^k+1}\oplus\cdots\oplus U_{j-k}^{2^{k+1}}\oplus U_{j-k}^{2^{k-1}}$$
$$\cdots$$
$$W_j=U_0^{2^j}\oplus U_0^{2^j+1}\oplus\cdots\oplus U_0^{2^{j+1}-1}$$

W_j 空间分解的子空间序列可写作 $U_{j-1}^{2^l+m}$，m=0,1,⋯,2^l-1；l=1,2,⋯。子空间序列 $U_{j-1}^{2^l+m}$ 的标准正交基为 $\{2^{-(j-1)/2}u_{2^l+m}(2^{j-l}t-k):k\in Z\}$。容易看出，当 l=0 和 m=0 时，子空间序列 $U_{j-1}^{2^l+m}$ 简化为 $U_j^1=W_j$，相应的正交基简化为 $2^{-j/2}u_1(2^{-j}t-k)=2^{-j/2}\psi(2^{-j}t-k)$，它恰好是标准正交小波族 $\{\psi_{j,k}(t)\}$。

若 n 是一个倍频程细化的参数，即令 $n=2^l+m$，则有小波包的简略记号 $\psi_{j,k,n}(t)=2^{-j/2}\psi_n(2^{-j}t-k)$，其中，$\psi_n(t)=2^{l/2}u_{2^l+m}(2^l t)$。把 $\psi_{j,k,n}(t)$ 称为有尺度指标 j、位置指标 k 和频率指标 n 的小波包。将它与前面的小波 $\psi_{j,k,}(t)$ 做比较可知，小波只有离散尺度 j 和离散平移 k 两个参数，而小波包除了这两个离散参数，还增加了一个频率参数 $n=2^l+m$。正是这个频率新参数的作用，使得小波包克服了小波时间分辨率高、时频率分辨率低的缺陷，于是，参数 n 表示 $\psi_n(t)=2^{l/2}u_{2^l+m}(2^l t)$ 函数的零交叉数目，也就是其波形的震荡次数。

定义（小波库）：由 $\psi_n(t)$ 生成的函数族 $\psi_{j,k,n}(t)$（其中 $n\in Z_+$；j，$k\in Z$）称为由尺度函数 $\psi(t)$ 构造的小波库。

推论：对于每个 j=0,1,2,…

$$L^2(R)=\bigoplus_{j\in Z}W_j=\cdots\oplus W_{-1}\oplus W_0\oplus U_0^2\oplus U_0^3\oplus\cdots$$

这时，

$\{u_{j,k},u_n(t-k)\,|\,j=\cdots,-1,0;\ n=2,3,\cdots$ 且 $k\in Z\}$ 是 $L^2(R)$ 的一个正交基。

随着尺度 j 的增大，相应正交小波基函数的空间分辨率越高，而其频率分辨率则越低，这正是正交小波基的一大缺陷。而小波包却具有随 j 增大而变宽的频谱窗口进一步分割变细的优良性质，从而克服了正交小波变换的不足。

小波包可以对 W_j 进一步分解，从而提高频率分辨率，是一种比多分辨分析更加精细的分解方法，具有更好的时频特性。

4．小波包算法

下面给出小波包的分解算法和重构算法。设 $g_j^n(t)\in U_j^n$，则 g_j^n 可表示为

$$g_j^n(t)=\sum_l d_l^{j,n}u_n(2^j t-l)$$

小波包分解算法由 $\{d_l^{j+1,n}\}$ 求 $\{d_l^{j,2n}\}$ 与 $\{d_l^{j,2n+1}\}$。

$$d_l^{j,2n}=\sum_k a_{k-2l}d_k^{j+1,n}$$

$$d_l^{j,2n+1}=\sum_k b_{k-2l}d_k^{j+1,n}$$

小波包重构算法由 $\{d_l^{j,2n}\}$ 求 $\{d_l^{j,2n+1}\}$ 与 $\{d_l^{j+1,n}\}$。

由于篇幅所限，这里就不再一一介绍了，感兴趣的读者可以参考相关文献。

13.1.5　几种常用的小波

下面介绍的 8 种常用的小波都包含在 MATLAB 小波工具箱中，请读者参看表 13-1。

表 13-1　MATLAB 对应的小波函数

参数表示	小波基的名称
morl	Morlet 小波
mexh	墨西哥小波
meyr	Meyer 小波
haar	Haar 小波
dbN	紧支撑正交小波
symN	近似对称的紧支撑双正交小波
coifN	Coifmant 小波
biorNr.Nd	双正交样条小波

1．Haar 小波

A.Haar 于 1990 年提出一种正交函数系，定义如下：

$$\psi_{\mathrm{H}}=\begin{cases}1 & 0\leqslant x<1/2\\ -1 & 1/2\leqslant x<1\\ 0 & \text{其他}\end{cases}$$

这是一种最简单的正交小波，即

$$\int_{-\infty}^{\infty}\psi(t)\psi(x-n)\mathrm{d}x=0 \qquad n=\pm1,\pm2,\cdots$$

2．Daubechies（dbN）小波系

Daubechies 小波是 Daubechies 从两尺度方程系数$\{h_k\}$出发设计出来的离散正交小波。一般简写为 dbN，N 是小波的阶数。小波 ψ 和尺度函数中的支撑区为 2N-1。ψ 的消失矩为 N。除 N＝1 外（Haar 小波），dbN 不具对称性。dbN 没有显式表达式（除 N＝1 外），但$\{h_k\}$的传递函数的模的平方有显式表达式。假设 $P(y)=\sum_{k=0}^{N-1}C_k^{N-1+k}y^k$，其中，$C_k^{N-1+k}$ 为二项式的系数，则有

$$\left|m_0(\omega)\right|^2=(\cos^2\frac{\omega}{2})^N P(\sin^2\frac{\omega}{2})$$

其中，$m_0(\omega)=\frac{1}{\sqrt{2}}\sum_{k=0}^{2N-1}h_k\mathrm{e}^{-ik\omega}$ 。

3．Biorthogonal（biorNr.Nd）小波系

Biorthogonal 函数系的主要特征体现在具有线性相位性，它主要应用在信号与图像的重

构中。通常的用法是采用一个函数进行分解，用另外一个小波函数进行重构。Biorthogonal 函数系通常表示为 biorNr.Nd 的形式：

```
Nr=1    Nd=1，3，5
Nr=2    Nd=2，4，6，8
Nr=3    Nd=1，3，5，7，9
Nr=4    Nd=4
Nr=5    Nd=5
Nr=6    Nd=8
```

其中，r 表示重构，d 表示分解。

4．Coiflet（coifN）小波系

Coiflet 函数也是由 Daubechies 构造的一个小波函数，它具有 coifN（N=1,2,3,4,5）这一系列，Coiflet 具有比 dbN 更好的对称性。从支撑长度的角度看，coifN 具有和 db3N 及 sym3N 相同的支撑长度；从消失矩的数目来看，coifN 具有和 db2N 及 sym2N 相同的消失矩数目。

5．SymletsA（symN）小波系

Symlets 函数系是由 Daubechies 提出的近似对称的小波函数，它是对 db 函数的一种改进。Symlets 函数系通常表示为 symN（N=2,3,…,8）的形式。

6．Morlet（morl）小波

Morlet 函数定义为 $\Psi(x)=C\mathrm{e}^{-x^2/2}\cos 5x$，它的尺度函数不存在，且不具有正交性。

7．Mexican Hat（mexh）小波

Mexican Hat 函数为

$$\Psi(x)=\frac{2}{\sqrt{3}}\pi^{-1/4}(1-x^2)\mathrm{e}^{-x^2/2}$$

它是 Gauss 函数的二阶导数，因为它像墨西哥帽的截面，所以有时称这个函数为墨西哥帽函数。墨西哥帽函数在时域与频域都有很好的局部化，并且满足

$$\int_{-\infty}^{\infty}\psi(x)\mathrm{d}x=0$$

由于它的尺度函数不存在，所以不具有正交性。

8．Meyer 函数

Meyer 小波函数 ψ 和尺度函数 φ 都是在频域中进行定义的，是具有紧支撑的正交小波。

$$\hat{\Psi}(\omega)=\begin{cases}(2\pi)^{-1/2}\mathrm{e}^{j\omega/2}\sin\left(\frac{\pi}{2}\nu\left(\frac{3}{2\pi}|\omega|-1\right)\right) & \frac{2\pi}{3}\le|\omega|\le\frac{4\pi}{3}\\(2\pi)^{-1/2}\mathrm{e}^{j\omega/2}\cos\left(\frac{\pi}{2}\nu\left(\frac{3}{2\pi}|\omega|-1\right)\right) & \frac{4\pi}{3}\le|\omega|\le\frac{8\pi}{3}\\0 & |\omega|\notin\left[\frac{2\pi}{3},\frac{8\pi}{3}\right]\end{cases}$$

其中，$\nu(a)$ 为构造 Meyer 小波的辅助函数，且有

$$\hat{\phi}(\omega)=\begin{cases}(2\pi)^{-1/2} & |\omega|\le\frac{2\pi}{3}\\ (2\pi)^{-1/2}\cos(\frac{\pi}{2}\nu(\frac{3}{2\pi}|\omega|-1)) & \frac{2\pi}{3}\le\omega\le\frac{4\pi}{3}\\ 0 & |\omega|>\frac{4\pi}{3}\end{cases}$$

13.1.6　小波应用演示示例

前面几节分别介绍了小波分析基本的数学理论，本节将向读者介绍有关小波应用的演示示例，以便将读者从抽象的理论中脱离出来。

例 13-1，含突变点信号 freqbrk 的连续小波变换 mesh 图。

MATLAB 程序设置如下：

```
load freqbrk
s=freqbrk;
coefs=cwt(s,[1;30],'morl');
colormap([0 0 0])
mesh(abs(coefs));
```

结果如图 13-3 所示。从图中可以清晰地看出突变点发生在 500 点处。

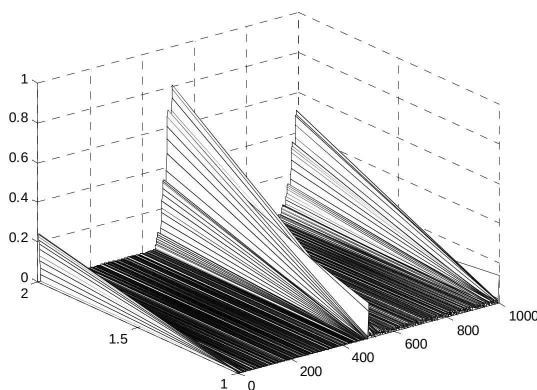


图 13-3　连续小波变换 mesh 图

例 13-2，突变点信号 freqbrk 的离散小波变换。

MATLAB 程序设置如下：

```
load freqbrk
s=freqbrk;
ls=length(s);
[C,L]=wavedec(s,4,'db5');
A4=wrcoef('a',C,L,'db5',4);
D4=wrcoef('d',C,L,'db5',1);
subplot(3,1,1);plot(abs(s),'k');
subplot(3,1,2);plot(abs(A4),'k');
subplot(3,1,3);plot(abs(D4),'k');
```

结果如图 13-4 所示。从图中可以看出，其检测结果与图 13-3 一致，但离散小波变换更

易于实际应用。

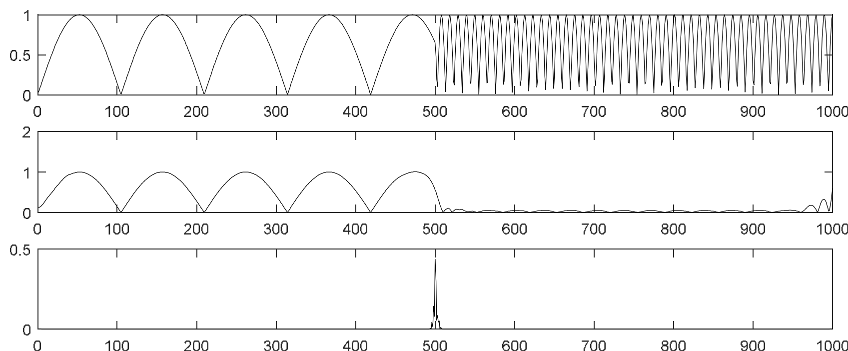


图 13-4　离散小波变换

13.2　小波工具箱介绍

13.1 节主要介绍了有关小波分析的基本知识，本节介绍关于小波工具箱的操作，使用户能从感性的视觉角度来学习小波工具箱。

13.2.1　小波工具箱的启动

启动小波工具箱有两种方法：第一种方法是在 MATLAB 窗口中输入 wavemenu 命令；第二种方法是在应用程序标签下单击 Wavelet Design & Analysis 按钮，如图 13-5 所示。

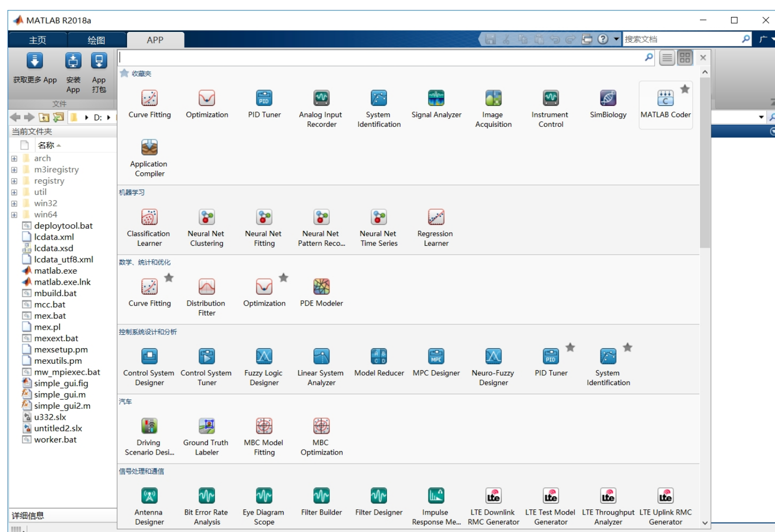


图 13-5　小波工具箱启动

按照上述两种方法操作，会弹出如图 13-6 所示的小波工具箱主界面。

在小波工具箱主界面中分别有一维工具、二维工具、三维工具、一维多重工具、显示工

具、一维专用工具、二维专用工具、小波设计工具及扩展工具等。

每种工具中又分为各种具体类型的应用，例如一维工具箱中包含一维小波、一维小波包、一维连续小波及一维复连续小波。由于篇幅所限，不再详细介绍这些工具。

下面介绍几个常见的小波工具。

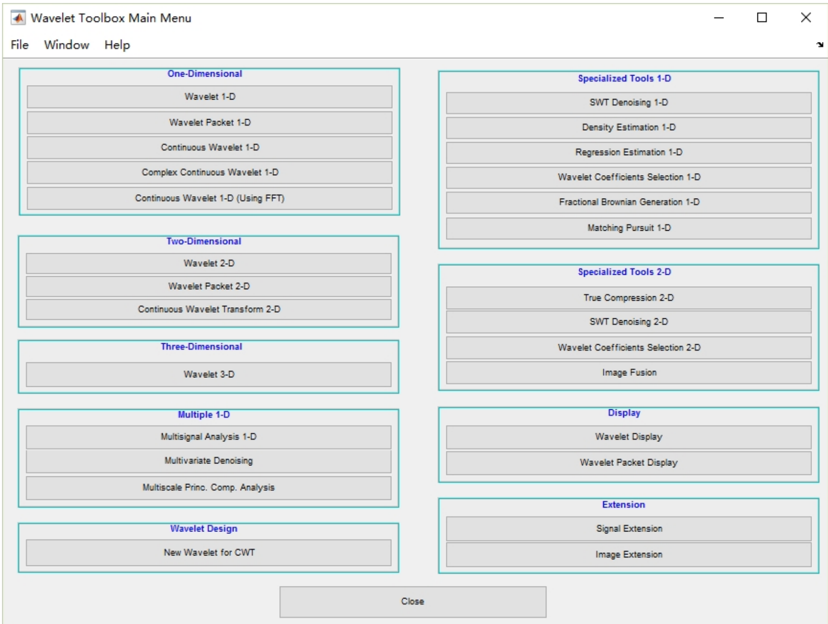


图 13-6　小波工具箱主界面

13.2.2　一维连续小波分析工具

读者通过对连续小波工具模块的学习后，利用小波工具箱软件便可以轻松地解决实际问题。在这部分中，读者将学习以下几点：

- 加载信号。
- 对信号进行连续小波变换。
- 生成小波系数图示。
- 生成给定尺度的小波系数图示。
- 生成局部小波系数图示。
- 选择所显示的图。
- 从尺度转换到伪频率信息上。
- 细节放大。
- 显示标准与绝对系数。
- 选择尺度。

下面向读者介绍有关的具体操作步骤。

例 13-3，利用一维连续小波分析工具分析正弦曲线噪声信号。

具体操作步骤如下：

[1] 打开一维连续小波分析工具。首先，在窗口中输入 wavemenu，弹出小波工具箱主界面，如图 13-7 所示。

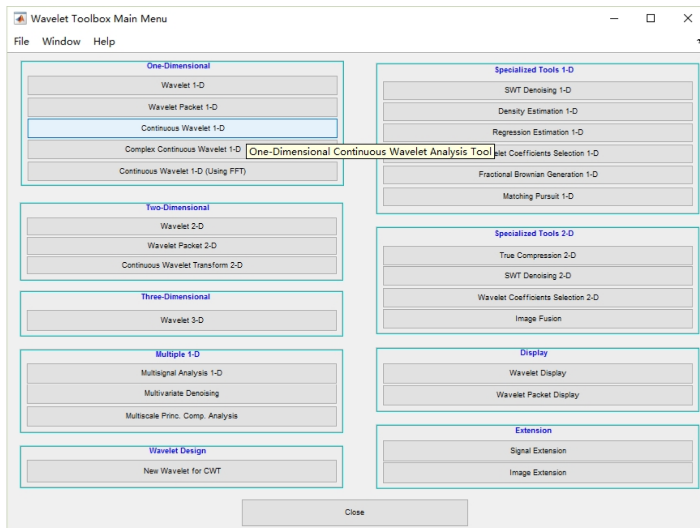


图 13-7　小波工具箱主界面

[2] 单击 Continuous Wavelet 1-D 工具按钮，弹出一维连续小波的主界面，如图 13-8 所示。

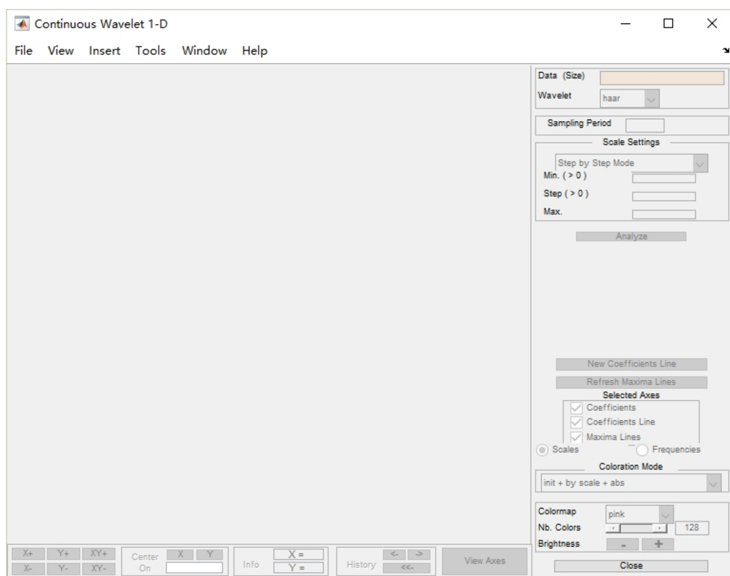


图 13-8　一维连续小波的主界面

[3] 下载信号源。选择 File→Load Signal 命令，在弹出的对话框中选择 noissin.mat 文件，它的路径为 toolbox/wavelet/wavedemo，单击 OK 按钮，这样信号就被下载到了此工具中，如图 13-9 所示。

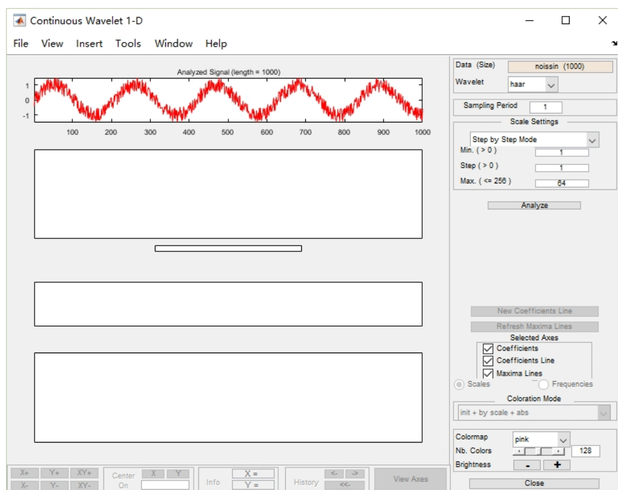


图 13-9　加载信号

[4] 执行连续小波变换。在如图 13-10 所示的界面中选择 db4 小波，并且尺度设置为 1～48。

[5] 单击 Analyze 按钮。经过短暂的计算后，工具显示对应尺度 a=24 的系数图和最大尺度图，如图 13-11 所示。

[6] 观看小波系数行。用鼠标右键单击系数图，如图 13-12 所示，在弹出的快捷菜单中选择 New Coefficients Line 选项，观察 Coefficients Line 图的变化。

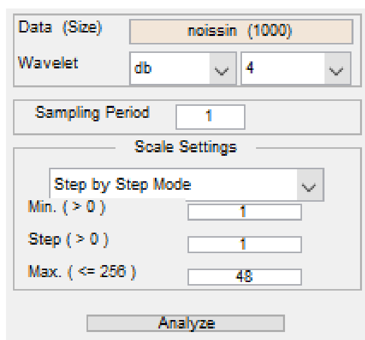


图 13-10　小波类型图

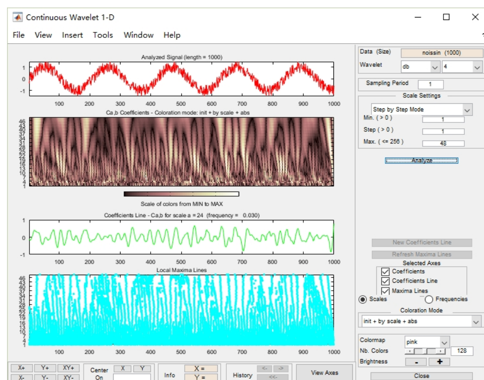


图 13-11　小波变换系数

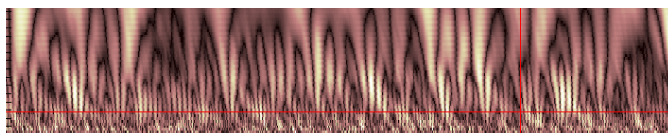


图 13-12　小波系数行观测

[7] 观看最大系数行，如图 13-13 所示。用鼠标右键单击系数图，在弹出的快捷菜单中选择 Refresh Maxima Line 命令。

在系数图中，鼠标移动的位置信息显示在工具屏幕下的 Info 文本框中，如图 13-14 所示。

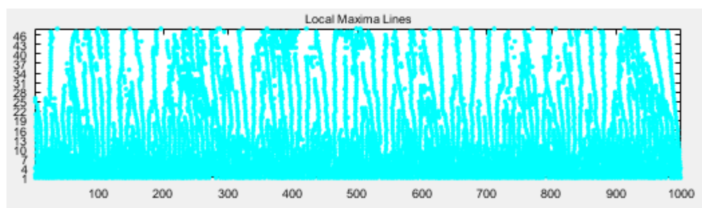


图 13-13 最大系数行观测

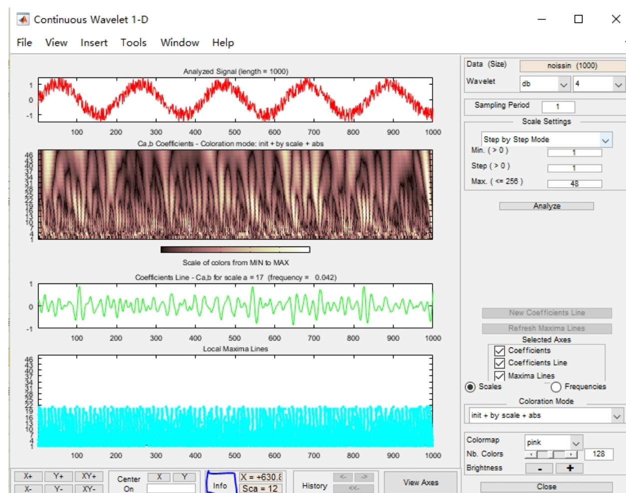


图 13-14 鼠标移动的位置信息

[8] 尺度转换到伪频率。在工具屏幕的右下方，选中 Frequencies 单选按钮，再次用鼠标右键单击系数图，如图 13-15 所示。

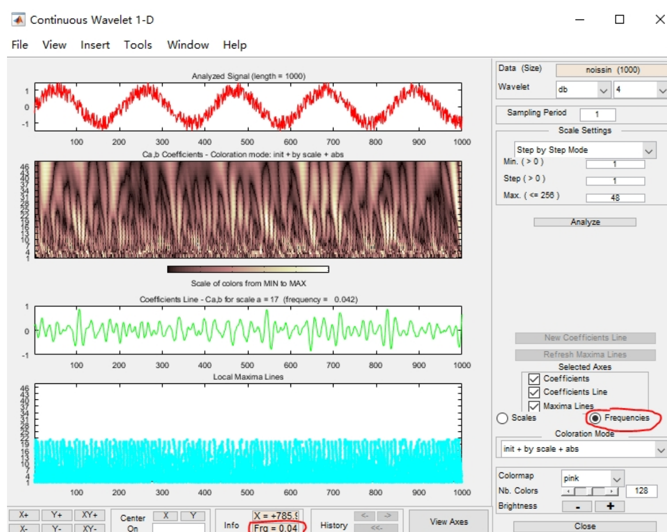


图 13-15 频率显示

[9] 在 Selected Axes 下取消勾选后两个复选框，如图 13-16 所示。

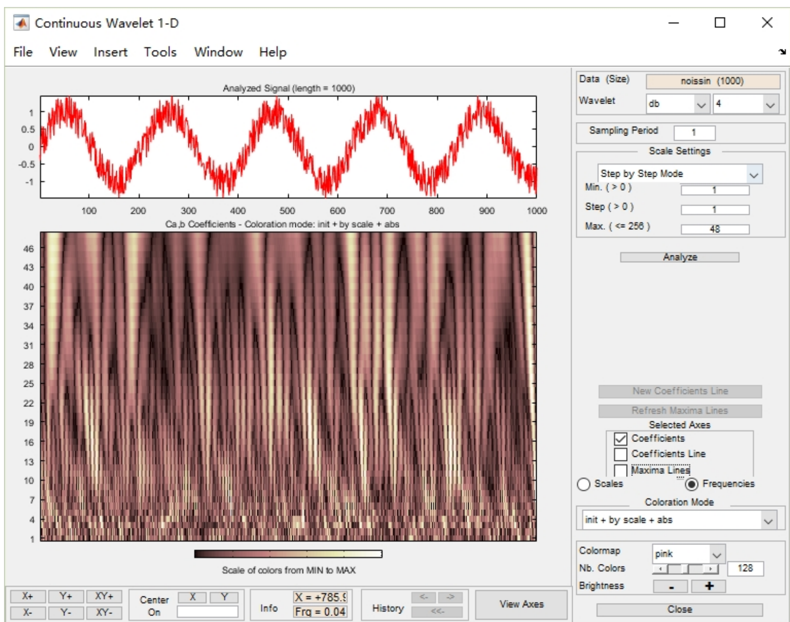


图 13-16　图形数目显示

[10] 放大细节。通过画矩形框得到的图形如图 13-17 所示。

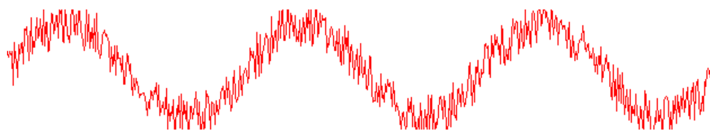


图 13-17　放大细节

[11] 单击“X+”按钮，如图 13-18 所示，显示水平放大后的信号，如图 13-19 所示。

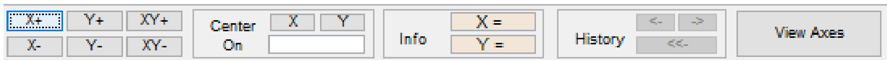


图 13-18　放大键

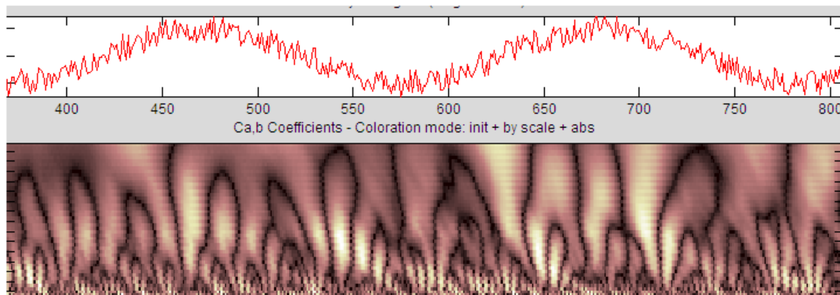


图 13-19　水平放大

[12] 观察标准系数图与绝对系数图。在 Coloration Mode 下拉列表框中分别选中 absolute modes 和 normal modes 选项，结果如图 13-20 所示。

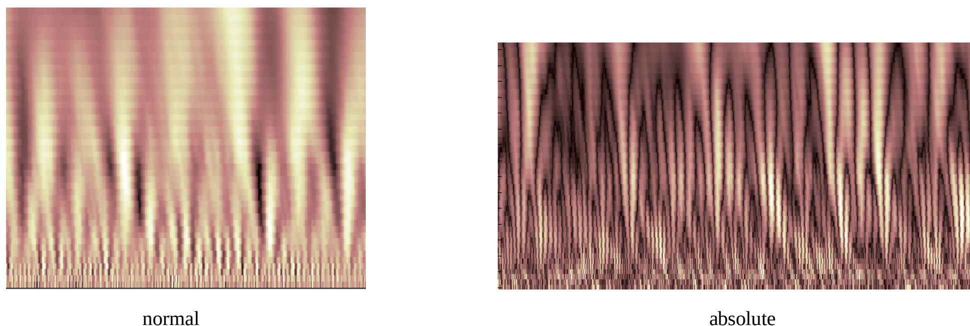


图 13-20 两种模式系数显示

上述过程也可以通过 MATLAB 命令实现，具体代码设置如下：

```
load noissin;
whos
c = cwt(noissin,1: 48,'db4');
c1 = cwt(noissin,1: 48,'db4','plot');
c2 = cwt(noissin,2: 2: 128,'db4','plot');
```

13.2.3 一维离散小波分析工具

13.2.2 节向读者介绍了一维连续小波变换的例子，本节将向读者介绍利用小波工具箱解决一维离散小波的应用示例。在这部分中，读者将学习以下几点：

- 加载信号。
- 对信号进行一级小波分解。
- 近似重构系数。
- 显示近似度与细节。
- 通过小波反变换重构信号。
- 对信号进行多级小波分解。
- 提取近似系数与细节。
- 重构 3 级近似度。
- 重构 1、2、3 级细节。
- 显示多级分解的结果。
- 重构 3 级分解中的初始信号。
- 去噪。
- 提纯分析。
- 压缩信号。
- 显示信号的统计值及直方图。

下面向读者介绍有关的具体操作步骤。

例 13-4，利用一维离散小波分析工具分析真实世界的噪声信号。

具体操作步骤如下：

[1] 打开一维小波分析工具。首先在窗口中输入 wavemenu，弹出小波工具箱主界面，

然后选择 Wavelet 1-D 选项，如图 13-21 所示。

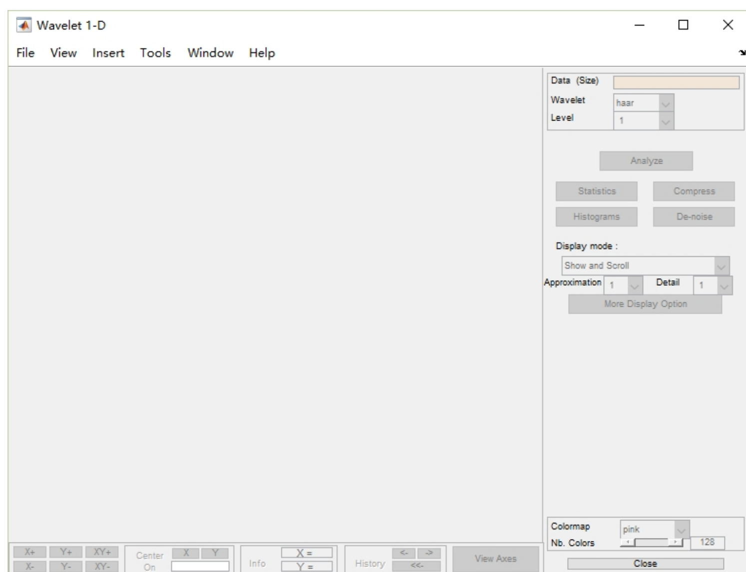


图 13-21　Wavelet 1-D 界面

[2] 下载信号源。选择 File→Load→Signal 命令，在弹出的对话框中选择 leleccum.mat 文件，它的路径为 toolbox/wavelet/wavedemo，单击 OK 按钮。这样，信号就被下载到了此工具中，如图 13-22 所示。

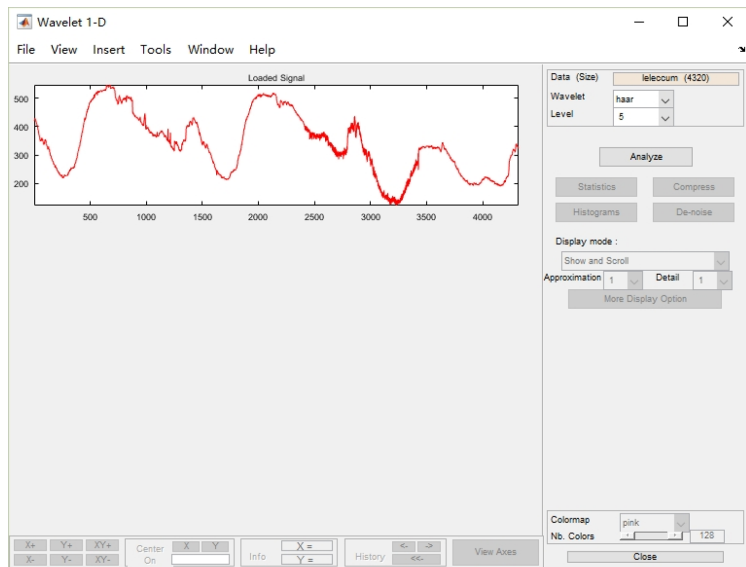


图 13-22　下载信号源

[3] 执行一级小波系数分解。在如图 13-22 所示的界面中选择 db1 小波进行一级分解，如图 13-23 所示。同时单击 Analyze 按钮，经过短暂的计算后，工具显示分解图，如图 13-24 所示。

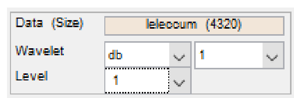


图 13-23 分解级数框图

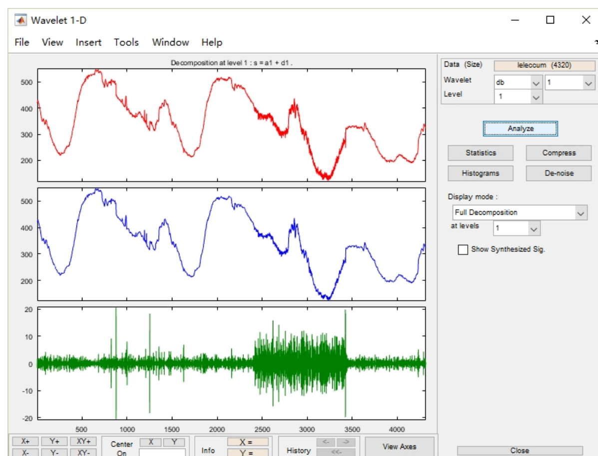


图 13-24 一级分解系数图

[4] 放大相关细节。画矩形框，如图 13-25 所示。然后单击“X+”按钮，显示水平放大后的信号，如图 13-26 所示。

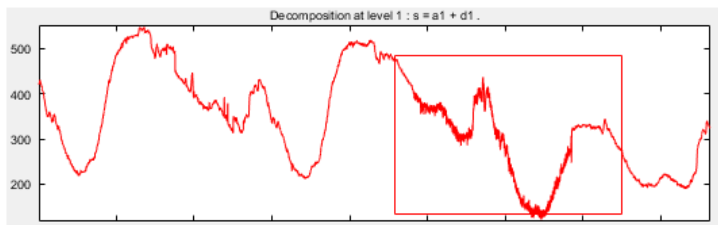


图 13-25 矩形框

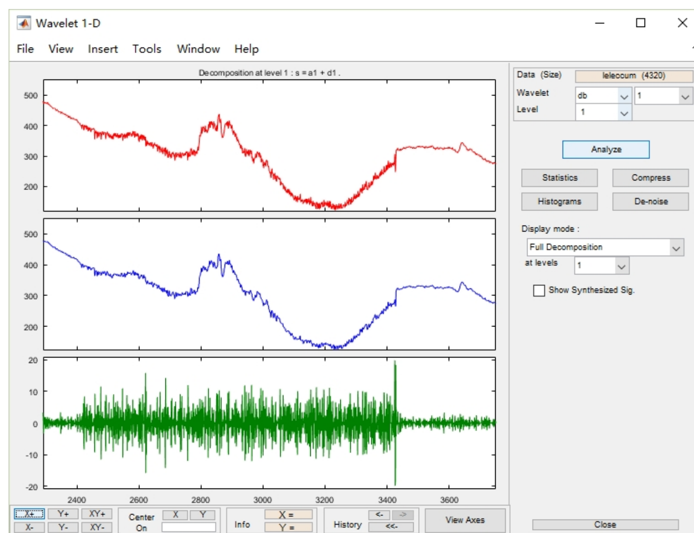


图 13-26 水平放大效果图

[5] 执行多级小波系数分解。在界面的右上方，将 Level 设置为 3，并单击 Analyze 按钮，如图 13-27 所示。这里，界面工具右下方提供了 5 种分解模式选项，如图 13-28 所示，读者可以分别选中观看不同的分解系数模式。

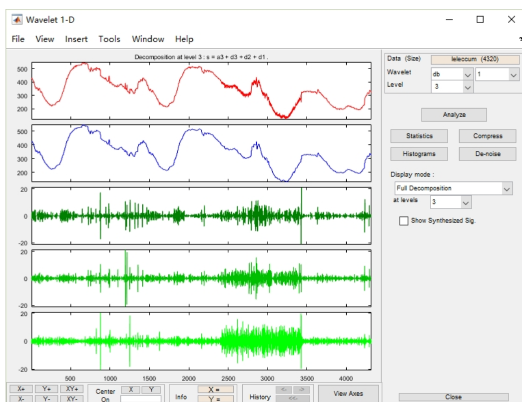


图 13-27　多级小波系数分解图

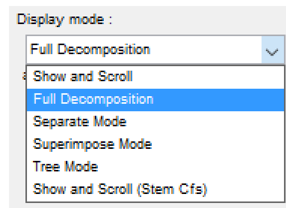


图 13-28　分解模式选项

[6] 去噪。单击 De-noise 按钮，如图 13-29 所示，弹出一维小波降噪窗口，如图 13-30 所示。

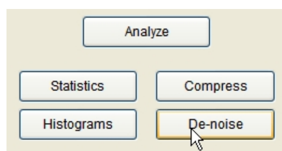


图 13-29　单击 De-noise 按钮

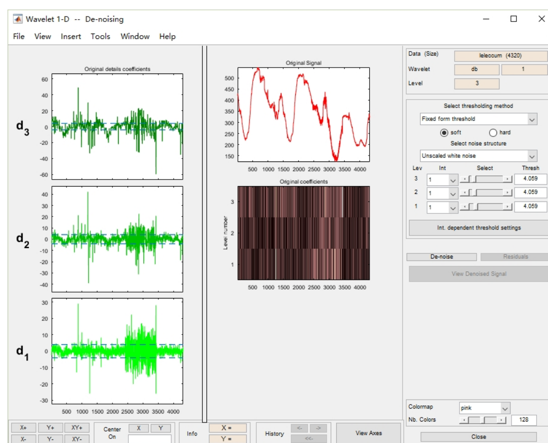


图 13-30　降噪窗口

[7] 提纯分析。将 Level 设置为 5， Wavelet 设置为 db3，单击 Analyze 按钮，如图 13-31 所示。

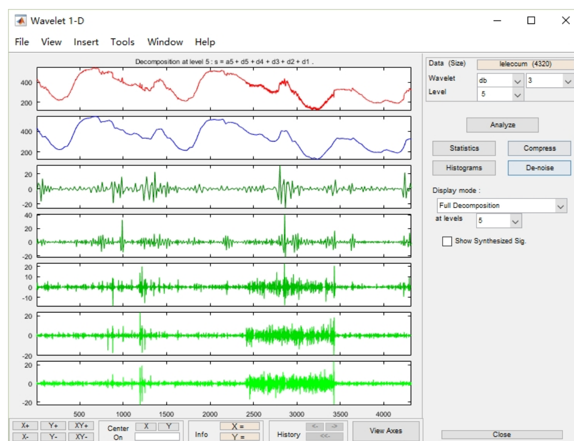


图 13-31　提纯分析

[8] 压缩信号。单击 Compress 按钮，弹出压缩窗口，如图 13-32 所示。

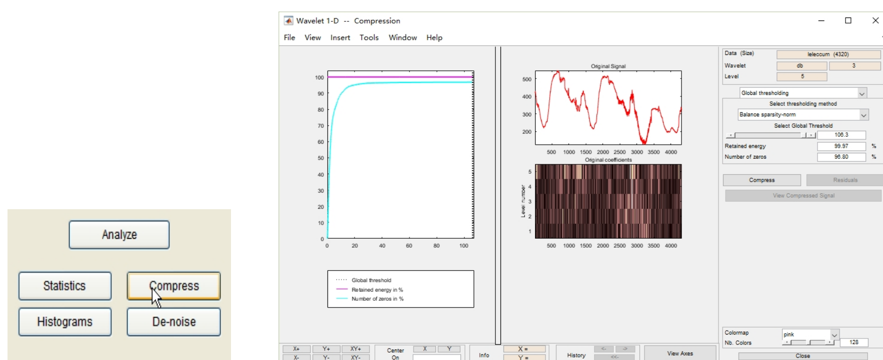


图 13-32　压缩界面

[9] 显示统计值。在 Wavelet 1-D 工具中单击 Statistics 按钮，弹出初始信号的统计值，如图 13-33 所示。单击 Histograms 按钮，显示如图 13-34 所示的直方图。

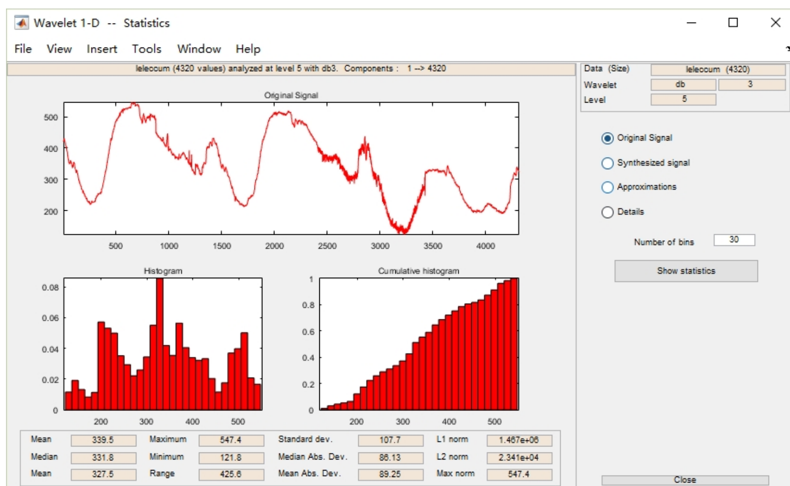


图 13-33　统计值

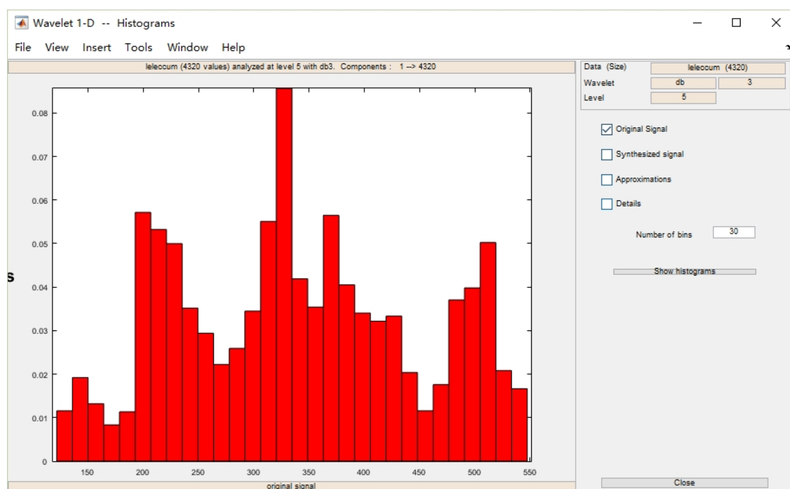


图 13-34　直方图

上述过程也可以通过 MATLAB 命令实现，具体代码设置如下：

```
load leleccum;
s = leleccum(1：3920);
l_s = length(s);
[cA1,cD1] = dwt(s,'db1');
A1 = upcoef('a',cA1,'db1',1,l_s);
D1 = upcoef('d',cD1,'db1',1,l_s);
subplot(1,2,1); plot(A1);
title('Approximation A1')
subplot(1,2,2); plot(D1);
title('Detail D1')
A0 = idwt(cA1,cD1,'db1',l_s);
err = max(abs(s-A0))
[C,L] = wavedec(s,3,'db1');
cA3 = appcoef(C,L,'db1',3);
A3 = wrcoef('a',C,L,'db1',3);
D1 = wrcoef('d',C,L,'db1',1);
D2 = wrcoef('d',C,L,'db1',2);
D3 = wrcoef('d',C,L,'db1',3);
subplot(2,2,1); plot(A3);
title('Approximation A3')
subplot(2,2,2); plot(D1);
title('Detail D1')
subplot(2,2,3); plot(D2);
title('Detail D2')
subplot(2,2,4); plot(D3);
title('Detail D3')
A0 = waverec(C,L,'db1');
err = max(abs(s-A0))
subplot(2,1,1);plot(s);title('Original'); axis off
subplot(2,1,2);plot(A3);title('Level 3 Approximation');
subplot(3,1,1); plot(D1);
title('Detail Level 1'); axis off
subplot(3,1,2); plot(D2);
title('Detail Level 2'); axis off
subplot(3,1,3); plot(D3);
title('Detail Level 3'); axis off
```

13.3　小波分析在图像处理中的应用

基于离散余弦变换的图像压缩算法，其基本思想是在频域对信号进行分解，去除信号点之间的相关性，并找出重要系数，滤掉次要系数，以达到压缩的效果。但该方法在处理过程中并不能提供时域的信息，在比较关心时域特性的时候显得无能为力。

但是这种应用的需求是很广泛的，比如遥感测控图像，要求在整幅图像有很高压缩比的同时，对热点部分的图像要有较高的分辨率。例如医疗图像，需要对某个局部的细节部分有很高的分辨率，单纯的频域分析的方法显然不能达到这个要求。虽然可以通过对图像进行分块分解，然后对每块作用不同的阈值或掩码来达到这个要求，但分块大小相对固定，有失灵活性。

在这个方面，小波分析就优越得多。由于小波分析固有的时频特性，可以在时频两个方

向对系数进行处理，这样就可以对感兴趣的部分提供不同的压缩精度。

13.3.1　基于小波变换的图像局部压缩

下面利用小波变换的时频局部化特性，举一个局部压缩的例子，读者可以通过这个例子看出小波变换应用于这类问题上的优越性。

例 13-5，图像局部压缩。

在本例中，把图像中部的细节系数都置零，从压缩图像中可以很明显地看出只有中间部分变得模糊（比如在原图中很清晰的围巾的条纹不能分辨），而其他部分的细节信息仍然可以分辨得很清楚。另外，小波域的系数表示的是原图像各频率段的细节信息，并且提供了一种位移相关的信息表述方式，可以通过对局部细节系数处理来达到局部压缩的效果。

MATLAB 代码设置如下：

```
load tire
%使用 sym4 小波对信号进行一层小波分解
[ca1,ch1,cv1,cd1]=dwt2(X,'sym4');
codca1=wcodemat(ca1,192);
codch1=wcodemat(ch1,192);
codcv1=wcodemat(cv1,192);
codcd1=wcodemat(cd1,192);
%将 4 个系数图像组合为一个图像
codx=[codca1,codch1,codcv1,codcd1]
%复制原图像的小波系数
rca1=ca1;
rch1=ch1;
rcv1=cv1;
rcd1=cd1;
%将 3 个细节系数的中部置零
rch1(33：97,33：97)=zeros(65,65);
rcv1(33：97,33：97)=zeros(65,65);
rcd1(33：97,33：97)=zeros(65,65);
codrca1=wcodemat(rca1,192);
codrch1=wcodemat(rch1,192);
codrcv1=wcodemat(rcv1,192);
codrcd1=wcodemat(rcd1,192);
%将处理后的系数图像组合为一个图像
codrx=[codrca1,codrch1,codrcv1,codrcd1]
%重建处理后的系数
rx=idwt2(rca1,rch1,rcv1,rcd1,'sym4');
subplot(221);
image(wcodemat(X,192)),colormap(map);title('原始图像');
subplot(222);
image(codx),colormap(map);title('一层分解后各层系数图像');
subplot(223);
image(wcodemat(rx,192)),colormap(map);title('压缩图像');
subplot(224);image(codrx),colormap(map);title('处理后各层系数图像');
%求压缩信号的能量成分
per=norm(rx)/norm(X)
per =1.0000
%求压缩信号与原信号的标准差
err=norm(rx-X)
```

误差结果：

```
err =
  315.1690
```

运行结果如图 13-35 所示。

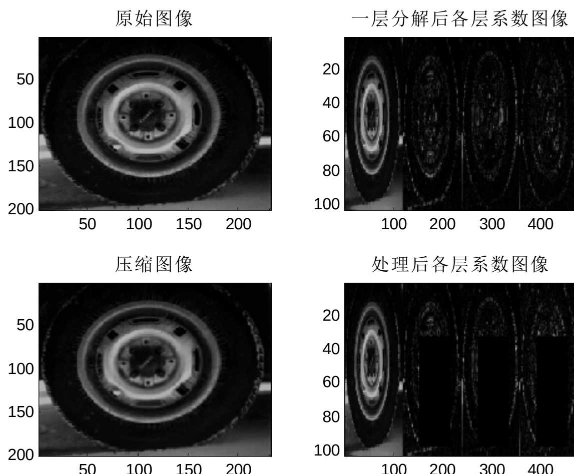


图 13-35　利用小波变换的局部压缩图像

最后需要说明的是，本例只是为了演示小波分析应用在图像局部压缩的方法。在实际应用中，可能不会只做一层变换，而且阈值的作用可能也不会是将局部细节系数全部清除，更一般的情况是在 N 层变换中通过选择零系数比例或能量保留成分作用不同的阈值，实现分片的局部压缩。而且，作用的阈值可以是方向相关的，即在 3 个不同方向的细节系数上作用不同的阈值。

13.3.2　二维小波变换的图像压缩

二维小波分析用于图像压缩是小波分析应用的一个重要方面。它的特点是压缩比高，压缩速度快，压缩后能保持图像的特征基本不变，且在传递过程中可以抗干扰。小波分析用于图像压缩具有明显的优点。

基于小波分析的图像压缩方法很多，比较成功的有小波包、小波变换零树压缩、小波变换矢量量化压缩等。

下面给出一个图像信号（即一个二维信号），利用二维小波分析对图像进行压缩。一幅图像做小波分解后，可得到一系列不同分辨率的子图像，不同分辨率的子图像对应的频率是不相同的。高分辨率（即高频）子图像上大部分点的数值接近于 0，越是高频这种现象越明显。对一幅图像来说，表现一幅图像最主要的部分是低频部分，所以一个最简单的压缩方法是利用小波分解，去掉图像的高频部分而只保留低频部分。图像压缩可按如下程序进行处理。

例 13-6，利用二维小波分析进行图像压缩。MATLAB 代码设置如下：

```
%装入图像
```

```
load tire
%显示图像
subplot(221);image(X);colormap(map)
title('原始图像');
axis square
disp('压缩前图像 X 的大小：');
whos('X')
%对图像用 bior3.7 小波进行二层小波分解
[c,s]=wavedec2(X,2,'bior3.7');
%提取小波分解结构中第一层低频系数和高频系数
ca1=appcoef2(c,s,'bior3.7',1);
ch1=detcoef2('h',c,s,1);
cv1=detcoef2('v',c,s,1);
cd1=detcoef2('d',c,s,1);
%分别对各频率成分进行重构
a1=wrcoef2('a',c,s,'bior3.7',1);
h1=wrcoef2('h',c,s,'bior3.7',1);
v1=wrcoef2('v',c,s,'bior3.7',1);
d1=wrcoef2('d',c,s,'bior3.7',1);
c1=[a1,h1;v1,d1];
%显示分解后各频率成分的信息
subplot(222);image(c1);
axis square
title('分解后低频和高频信息');
%下面进行图像压缩处理
%保留小波分解第一层低频信息，进行图像的压缩
%第一层的低频信息即为 ca1，显示第一层的低频信息
%首先对第一层信息进行量化编码
ca1=appcoef2(c,s,'bior3.7',1);
ca1=wcodemat(ca1,440,'mat',0);
%改变图像的高度
ca1=0.5*ca1;
subplot(223);image(ca1);colormap(map);
axis square
title('第一次压缩');
disp('第一次压缩图像的大小为：');
whos('ca1')
%保留小波分解第二层低频信息，进行图像的压缩，此时压缩比更大
%第二层的低频信息即为 ca2，显示第二层的低频信息
ca2=appcoef2(c,s,'bior3.7',2);
%首先对第二层信息进行量化编码
ca2=wcodemat(ca2,440,'mat',0);
%改变图像的高度
ca2=0.25*ca2;
subplot(224);image(ca2);colormap(map);
axis square
title('第二次压缩');
disp('第二次压缩图像的大小为：');
whos('ca2')
```

输出结果如下所示。

压缩前图像 X 的大小：

```
Name       Size           Bytes  Class    Attributes
  X       200x232         371200  double
```

第一次压缩图像的大小：

```
Name       Size           Bytes  Class    Attributes
 ca1      107x123         105288  double
```

第二次压缩图像的大小：

```
Name      Size          Bytes  Class    Attributes
 ca2     61x69          33672  double
```

运行结果如图 13-36 所示。

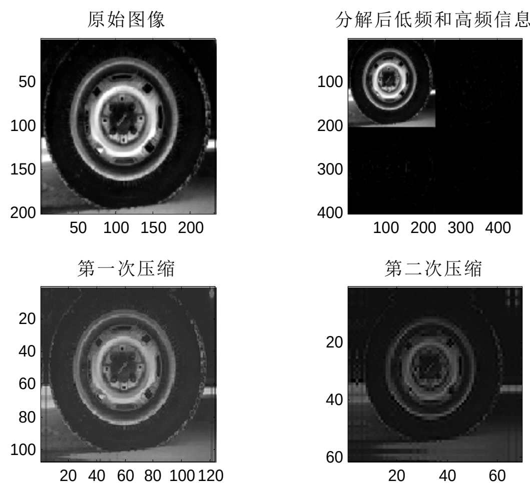


图 13-36　利用二维小波分析进行图像压缩

通过图像对比可以看出，第一次压缩提取的是原始图像中小波分解第一层的低频信息，此时压缩效果较好，压缩比较小（约为 1/3）；第二次压缩是提取第一层分解低频部分的低频部分（即小波分解第二层的低频部分），其压缩比较大（约为 1/12），压缩效果在视觉上也基本过得去。这是一种最简单的压缩方法，只保留原始图像中的低频信息，不经过其他处理即可获得较好的压缩效果。在上面的例子中，还可以只提取小波分解第 3、4、……层的低频信息。从理论上说，可以获得任意压缩比的压缩图像。

例 13-7，利用二维小波变换对图像进行压缩。MATLAB 代码设置如下：

```
%装入一个二维信号
load wbarb
%显示图像
subplot(121);image(X);colormap(map)
title('原始图像');
axis square
%下面进行图像压缩
%对图像用 db3 小波进行二层小波分解
[c,s]=wavedec2(X,2,'db3');
%使用 wavedec2 函数来实现图像的压缩
[thr,sorh,keepapp]=ddencmp('cmp','wv',X);
%输入参数中选择了全局阈值选项“gbl”，用来对所有高频系数进行相同的阈值量化处理
[Xcomp,cxc,lxc,perf0,perfl2]=wdencmp('gbl',c,s,'db3',2,thr,sorh,keepapp);
%将压缩后的图像与原始图像相比较，并显示出来
subplot(122);image(Xcomp);colormap(map)
title('压缩图像');
axis square
disp('小波分解系数中置 0 的系数个数百分比：');
```

```
perf0
disp('压缩后图像剩余能量百分比：');
perfl2
```

输出结果如下所示。

小波分解系数中置 0 的系数个数百分比：

```
perf0 =
  49.3839
```

压缩后图像剩余能量百分比：

```
perfl2 =
  99.9774
```

运行结果如图 13-37 所示。

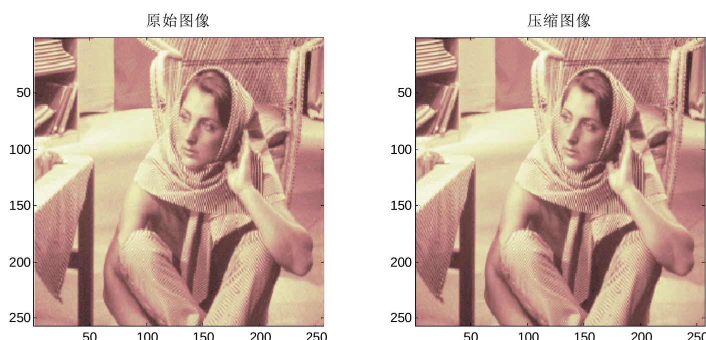


图 13-37　利用二维小波变换对图像进行压缩

利用二维小波变换进行图像压缩时，小波变换将图像从空间域变换到时间域，它的作用与以前在图像压缩中所用到的离散余弦变换（DCT）和傅里叶变换（FFT）等的作用类似。但是要很好地进行图像压缩，需要综合利用多种技术，特别是数据的编码与解码算法等，所以利用小波分析进行图像压缩通常需要利用小波分析和许多其他相关技术共同完成。

13.3.3　图像压缩中阈值的确定与作用命令

由于阈值处理只关心系数的绝对值，并不关心系数的位置，所以二维小波变换系数的阈值化方法同一维情况大同小异。为了方便用户使用，小波工具箱对某些阈值化方法提供了专门的二维处理命令。

下面通过一个例子来说明二维信号的小波压缩的一般方法，在这个例子中同时采用求默认阈值的 ddencmp 命令和基于经验公式的 wdcbm2 命令对图像进行压缩，并对压缩效果进行比较。

例 13-8，图像压缩中阈值的确定。MATLAB 代码设置如下：

```
load tire
%求得颜色映射表的长度，以便后面的转换
nbc=size(map,1);
%用默认方式求出图像的全局阈值
[thr,sorh,keepapp]=ddencmp('cmp','wv',X);
%对图像作用全局阈值
```

```
[xd,cxd,lxd,perf0,perfl2]=…
wdencmp('gbl',X,'bior3.5',3,thr,sorh,keepapp);
%用 bior.3.5 小波对图像进行三层分解
[c,s]=wavedec2(X,3,'bior3.5');
%指定 Birge-Massart 策略中的经验系数
alpha=1.5;
m=2.7*prod(s(1,: ));
%根据各层小波系数确定分层阈值
[thr1,nkeep1]=wdcbm2(c,s,alpha,m);
%对原图像作用分层阈值
[xd1,cxd1,sxd1,perf01,perfl21]=wdencmp('lvd',c,s,'bior3.5',3,thr1…,'s')
%将颜色映射表转换为灰度映射表
colormap(pink(nbc));
subplot(131);image(wcodemat(X,nbc));title('原始图像');
subplot(132);image(wcodemat(xd,nbc));
title('全局阈值化压缩图像');xlabel(['能量成分',num2str(perfl2),
…'%','零系数成分',num2str(perf0),'%']);
subplot(133);
image(wcodemat(xd1,nbc));
title('分层阈值化压缩图像');
xlabel(['能量成分',num2str(perfl21),'%',
…'零系数成分',num2str(perf01),'%']);
```

显示结果如图 13-38 所示。

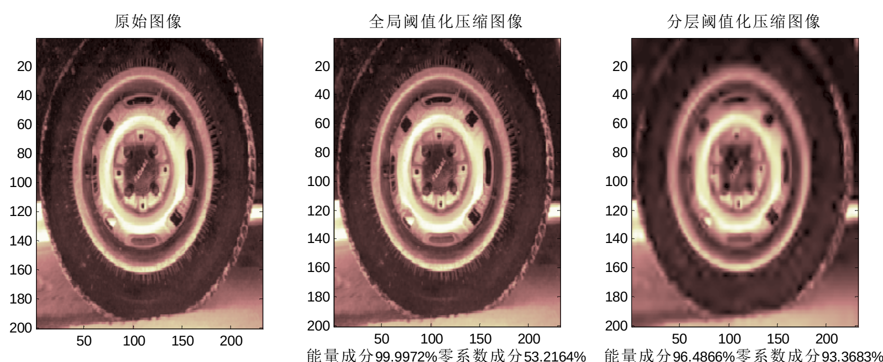


图 13-38　图像的全局阈值化压缩和分层阈值化压缩

可见，分层阈值化压缩方法同全局阈值化方法相比，在能量损失不是很大的情况下可以获得最高的压缩比，这主要是因为层数和方向相关的阈值化方法能利用更精细的细节信息进行阈值化处理。

13.3.4　基于小波包变换的图像压缩

小波分析之所以在信号处理中有着强大的功能，是基于其分离信息的思想，分离到各个小波域的信息排除了与其他小波域的关联，使得处理的时候更为灵活。全局阈值化方法作用的信息粒度太大，不够精细，所以很难同时获得高的压缩比和能量保留成分，在作用的分层阈值以后，性能明显提高，因为分层阈值更能体现信号固有的时频局部特性。

但是小波分解仍然不够灵活，分解出来的小波树只有一种模式，不能完全地体现时频局部化信息。而压缩的核心思想是尽可能去除各小波域系数之间的信息关联，最大限度地体现

时频局部化的信息。因此，实际的压缩算法多采用小波包算法，而小波树的确定则是根据不同的信息论准则，以使分解系数表达的信息密度达到最高。

下面通过一个例子来说明小波包分析在图像压缩中的应用，并给出性能参数，以便同基于小波分析的压缩进行比较。

例 13-9，小波包分析的图像压缩。MATLAB 代码设置如下：

```
%读入信号
load detfingr
%求颜色索引表长度
nbc=size(map,1);
%得到信号的阈值，保留层数，小波树优化标准
[thr,sorh,keepapp,crit]=ddencmp('cmp','wp',X)
%通过以上得到的参数对信号进行压缩
[xd,treed,perf0,perfl2]=wpdencmp(X,sorh,4,'sym4',crit,thr*2,keepapp);
%更改索引表为 pink 索引表
colormap(pink(nbc));
subplot(121);image(wcodemat(X,nbc));title('原始图像');
subplot(122);image(wcodemat(xd,nbc));title('全局阈值化压缩图像');
xlabel(['能量成分',num2str(perfl2),'%',
…'零系数成分',num2str(perf0),'%']);
plot(treed);
```

得到的压缩结果如图 13-39 所示。

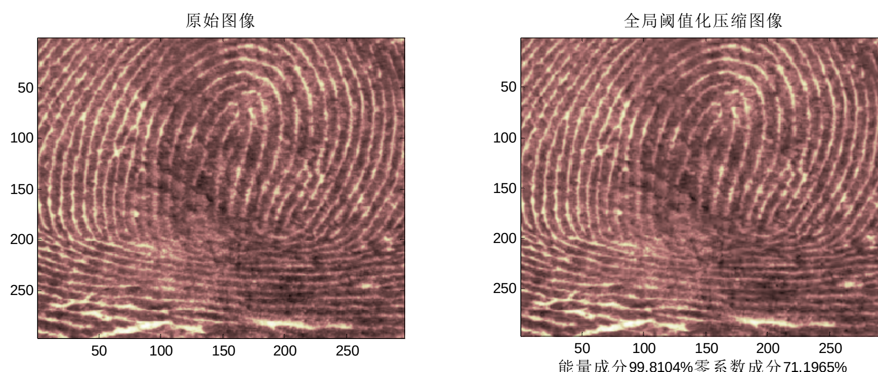


图 13-39　基于小波包分析的图像压缩

压缩过程中使用的最优小波树如图 13-40 所示。

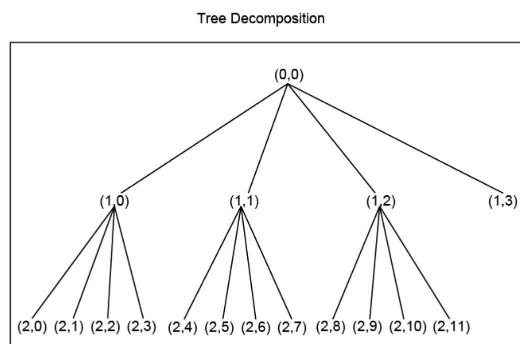


图 13-40　最优小波树

ddencmp 命令和 wdcbm2 命令是 MATLAB 小波工具箱提供的自动获取阈值和自动使用小波包压缩的命令，后者将分解阈值化和重建综合起来。在将小波包用于信号压缩的过程中，ddencmp 命令返回的最优小波树标准都是阈值化标准。根据这个标准确定的最优小波树可以使得压缩过程的零系数成分最高，并且自动降低计算量。

最后需要说明一点，对高频成分很多的图像，小波包分解细节信息的特点尤其能发挥其优势。正因为这一点，FBI 的指纹库采用的就是基于小波包的压缩算法 WSQ。

图像压缩是应用非常广泛的一类问题，所以其机器实现效率是至关重要的。在实际的应用中，如 JPEG2000，一般不采用通常的 Mallat 算法做小波分解，而是应用特定的双正交小波，利用其滤波器分布规则的特性，用移位操作来实现滤波操作。

13.3.5　小波变换用于图像去噪

噪声可以理解为妨碍人的视觉器官或系统传感器对所接收图像源进行理解或分析的各种因素。一般噪声是不可预测的随机信号，它只能用概率统计的方法去认识。

噪声影响图像处理的输入、采集、处理的各个环节及输出结果的全过程。特别是图像输入采集的噪声是一个十分关键的问题，若输入伴有较大噪声，必然影响处理全过程及输出结果。因此，一个良好的图像处理系统，不论是模拟处理还是计算机处理，无不把减少最前一级的噪声作为主攻目标。去噪已成为图像处理中极其重要的步骤。

对二维图像信号的去噪方法同样适用于一维信号，尤其是对于几何图像更适合。二维模型可以表述为

$$s(i,j)=f(i,j)+\delta\cdot e(i,j)\qquad i,j=0,1,\cdots,m-1$$

其中，e 是标准偏差不变的高斯白噪声。二维信号用二维小波分析的去噪步骤有 3 步：

（1）二维信号的小波分解。选择一个小波和小波分解的层次 N，然后计算信号 s 到第 N 层的分解。

（2）对高频系数进行阈值量化。对于 1～N 的每一层，选择一个阈值，并对这一层的高频系数进行软阈值量化处理。

（3）二维小波的重构。根据小波分解的第 N 层的低频系数和经过修改的从第一层到第 N 层的各层高频系数计算二维信号的小波重构。

在这 3 个步骤中，重点是如何选取阈值和阈值的量化。

下面给出一个二维信号，并利用小波分析对信号进行去噪处理。MATLAB 的去噪函数有 ddencmp、wdencmp 等。

例 13-10，去噪示例 1。MATLAB 代码设置如下：

```
%装入图像
load detfingr
%下面进行噪声的产生
init=3718025452;
rand('seed',init);
Xnoise=X+18*(rand(size(X)));
%显示原始图像及它含噪声的图像
colormap(map);
```

```
subplot(1,3,1);image(wcodemat(X,192));
title('原始图像X')
axis square
subplot(1,3,2);
image(wcodemat(X,192));
title('含噪声的图像Xnoise');
axis square
%用sym5小波对图像信号进行二层小波分解
[c,s]=wavedec2(X,2,'sym5');
%下面进行图像的去噪处理
%使用ddencmp函数来计算去噪的默认阈值和熵标准
%使用wdencmp函数来实现图像的压缩
[thr,sorh,keepapp]=ddencmp('den','wv',Xnoise);
[Xdenoise,cxc,lxc,perf0,perfl2]=
…wdencmp('gbl',c,s,'sym5',2,thr,sorh,keepapp);
%显示去噪后的图像
subplot(133);image(Xdenoise);
title('去噪后的图像');
axis square
```

输出结果如图 13-41 所示，从 3 幅图像的比较可以看出，MATLAB 中的 ddencmp 和 wdencmp 函数可以有效地进行去噪处理。

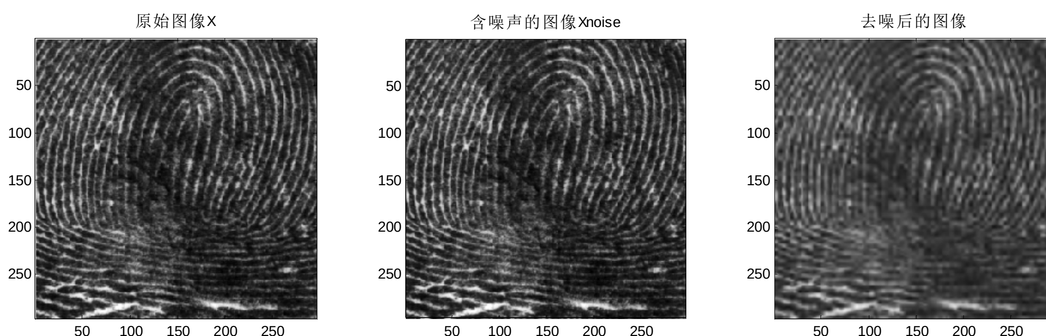


图 13-41　去噪示例 1

例 13-11，去噪示例 2。MATLAB 代码设置如下：

```
load julia
%画出原始图像
subplot(221);image(X);colormap(map);
title('原始图像');
axis square
%产生含噪图像
init=2055615866;randn('seed',init)
x=X+38*randn(size(X));
%画出含噪图像
subplot(222);image(x);colormap(map);
title('含噪声图像');
axis square;
%下面进行图像的去噪处理
%用小波函数sym4对x进行二层小波分解
[c,s]=wavedec2(x,2,'sym4');
%提取小波分解中第一层的低频图像，即实现了低通滤波去噪
a1=wrcoef2('a',c,s,'sym4');
%画出去噪后的图像
```

```
subplot(223);image(a1);
title('第一次去噪图像');
axis square;
%提取小波分解中第二层的低频图像，即实现了低通滤波去噪
%相当于把第一层的低频图像经过再一次的低频滤波处理
a2=wrcoef2('a',c,s,'sym4',2);
%画出去噪后的图像
subplot(224);image(a2);title('第二次去噪图像');
axis square;
```

输出结果如图 13-42 所示。

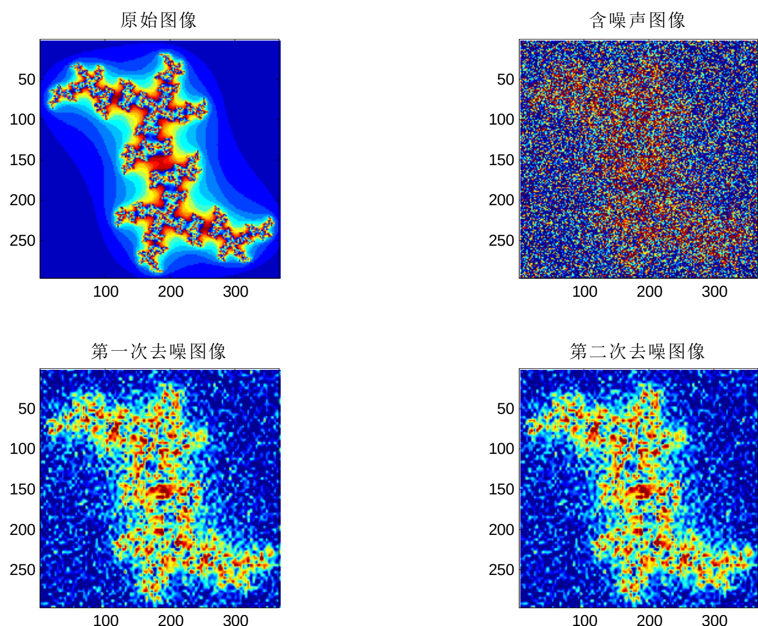


图 13-42　去噪示例 2

从上面的输出结果可以看出，第一次去噪已经滤去了大部分的高频噪声，但从去噪图像与原始图像的对比可以看出，第一次去噪后的图像中还是含有不少的高频噪声；第二次去噪是在第一次去噪的基础上，再次滤去其中的高频噪声，从去噪的结果可以看出，它具有较好的去噪效果。

下面再给出一个含有较少噪声的图像。由于原始图像中只含有较少的高频噪声，如果按照上一个例子把高频噪声全部滤掉，将损坏图像中固有的高频有用信号。因此这幅图像适合采用小波分解系数阈值量化方法进行去噪处理。

例 13-12，去噪示例 3。MATLAB 代码设置如下：

```
load wmandril
%画出原始图像
subplot(131);image(X);colormap(map);
title('原始图像');
axis square
%产生含噪声图像
init=2055615866;randn('seed',init)
x=X+10*randn(size(X));
```

```
%画出含噪声图像
subplot(132);image(X);colormap(map);
title('含噪声图像');
axis square
%下面进行图像的去噪处理
%用小波画数 coif3 对 x 进行二层小波分解
[c,s]=wavedec2(x,2,'coif3');
%提取小波分解中第一层的低频图像，即实现了低通滤波去噪
%设置尺度向量 n
n=[1,2]
%设置阈值向量 p
p=[10.12,23.28];
%对三个方向高频系数进行阈值处理
nc=wthcoef2('h',c,s,n,p,'s');
nc=wthcoef2('v',c,s,n,p,'s');
nc=wthcoef2('d',c,s,n,p,'s');
%对新的小波分解结构[nc,s]进行重构
xx=waverec2(nc,s,'coif3');
%画出重构后图像的波形
subplot(133);image(X);colormap(map);
title('去噪后的图像');
axis square
```

输出结果如图 13-43 所示。

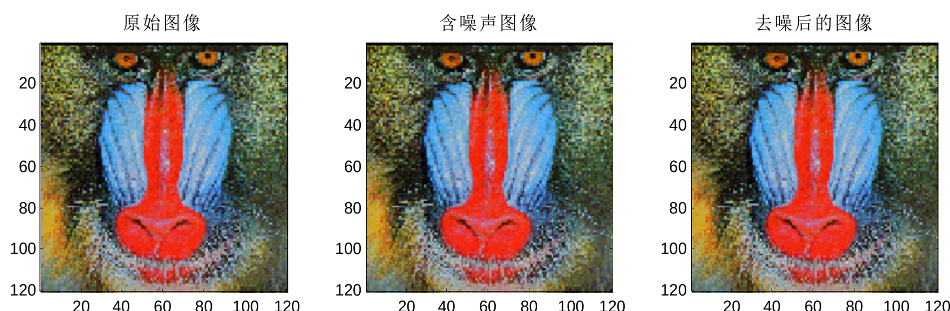


图 13-43　去噪示例 3

二维信号在应用中一般表现为图像信号，二维信号在小波域中的去噪方法的基本思想与一维情况一样。在阈值选择上，可以使用统一的全局阈值，又可以分为 3 个方向，分别是水平方向、竖直方向和对角方向。这样就可以把所有方向的噪声分离出来，通过作用阈值抑制其成分。

13.3.6　小波分析用于图像增强

小波分析在二维信号（图像）处理方面的优点主要体现在其时频分析特性上，前面介绍了基于这种特性的一些应用示例，但对二维信号小波系数的处理方法只介绍了阈值化方法，下面将介绍以前在一维信号中用到的抑制系数的方法，这种方法在图像处理领域主要应用于图像增强。

图像增强问题的基本目标是对图像进行一定的处理，使其结果比原图更适用于特定的应

用领域。这里“特定”这个词非常重要，因为几乎所有的图像增强问题都是与问题背景密切相关的，脱离了问题本身的知识，图像的处理结果可能并不一定适用。比如某种方法可能非常适用于处理 X 射线图像，但同样的方法不一定适用于处理火星探测图像。

在图像处理领域，图像增强问题主要通过时域（沿用信号处理的说法，空域可能对图像更适合）和频域处理两种方法来解决。时域方法通过直接在图像点上作用算子或掩码来解决，频域方法通过修改傅里叶变换系数来解决。这两种方法的优劣很明显，时域方法方便、快速，但会丢失很多点之间的相关信息；频域方法可以很详细地分离出点之间的关系，但需要做两次数量级为 nlogn 的傅里叶变换和逆变换的操作，计算量大得多。

小波分析是以上两种方法的权衡结果，建立在如下的认识基础上：傅里叶分析的所有点的分辨率都是原始图像的尺度，但对于问题本身的要求，可能不需要这么大的分辨率，而单纯的时域分析又显得太粗糙，小波分析的多尺度分析特性为用户提供了更灵活的处理方法。可以选择任意的分解层数，用尽可能少的计算量得到满意的结果。

小波变换将一幅图像分解为大小、位置和方向都不同的分量。在做逆变换之前可以改变小波变换域中某些系数的大小，这样就能够有选择地增加所感兴趣的分量而减少不需要的分量。下面给出一个图像增强的示例。

例 13-13，图像增强。MATLAB 代码设置如下：

```
load facets
%画出原始图像
subplot(121);image(X);colormap(map);
title('原始图像');
axis square
%下面进行图像的增强处理
%用小波函数 sym4 对 X 进行二层小波分解
[c,s]=wavedec2(X,2,'sym4');
sizec=size(c);
%对分解系数进行处理以突出轮廓部分，弱化细节部分
for i=1:size(2)
  if(c(i)>350)
    c(i)=2*c(i);
  else
    c(i)=0.5*c(i);
  end
end
%下面对处理后的系数进行重构
xx=waverec2(c,s,'sym4');
%画出重构后的图像
subplot(122);image(xx);
colormap(map);
title('增强图像');
axis square
```

输出结果如图 13-44 所示。

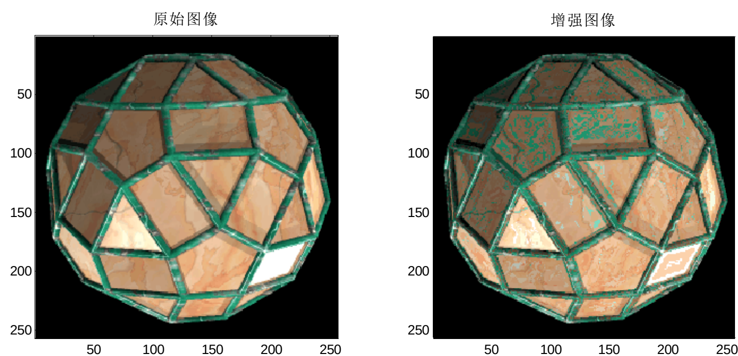


图 13-44 小波分析用于图像增强

13.3.7 图像钝化与锐化

图像钝化在时域中的处理相对简单，只需要对图像作用一个平滑滤波器，使得图像中的每个点与其相邻点做平滑即可，这里不做详细介绍。下面介绍一下基于傅里叶变换的频域处理方法。

下面以 chess 信号为例，通过两种方法对图像钝化的结果做一下比较。

例 13-14，采用 DCT 和小波方法来使图像钝化。MATLAB 代码设置如下：

```
%读入 chess 信号
load chess
%分别保存用 DCT 方法和小波方法的变换系数
blur1=X;
blur2=X;
%对原图像做二维离散余弦变换
ff1=dct2(X);
%对变换结果在频域做 BUTTERWORTH 滤波
for i=1:256
   for j=1:256
      ff1(i,j)=ff1(i,j)/(1+((i*j+j*j)/8192)^2);
   end
end
%重建变换后的图像
blur1=idct2(ff1);
%对图像做二层的二维小波分解
[c,l]=wavedec2(X,2,'db3');
csize=size(c);
%对低频系数进行放大处理，并抑制高频系数
for i=1:csize(2);
   if(c(i)>300)
      c(i)=c(i)*2;
   else
      c(i)=c(i)/2;
   end
end
%通过处理后的小波系数重建图像
blur2=waverec2(c,l,'db3');
%显示 3 幅图像
subplot(221);image(wcodemat(X,192));colormap(gray(256));
title('原始图像','fontsize',18);
```

```
subplot(223);image(wcodemat(blur1,192));colormap(gray(256));
title('采用 DCT 方法钝化图像','fontsize',18);
subplot(224);image(wcodemat(blur2,192));colormap(gray(256));
title('采用小波方法钝化图像','fontsize',18);
```

在命令行中运行该程序后，得到如图 13-45 所示的显示结果。

从图中可以看出，采用 DCT 在频域做滤波的方法得到的钝化结果更为平滑，这是因为其分辨率最高，而小波方法得到的结果在很多地方有不连续的现象，因为对系数做放大或抑制在阈值两侧有间断，而且分解层数很低，没有完全分离出频域的信息。同时，在做系数放大或抑制的时候，采用的标准是根据系数绝对值的大小，没有完全体现出其位置信息，但是在小波系数中，很容易在处理系数的过程中加入位置信息。

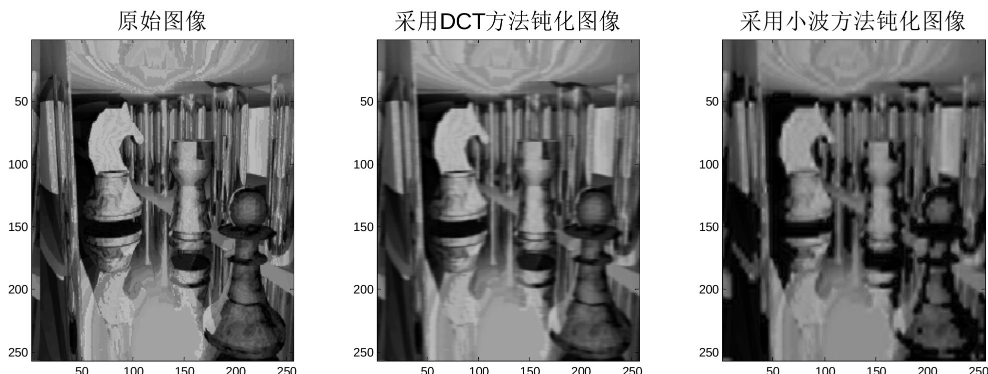


图 13-45　图像钝化

与图像钝化所做的工作相反，图像锐化的任务是突出高频信息，抑制低频信息，从快速变化的成分中分离出标识系统特性或区分子系统边界的成分，以便进行进一步的识别、分割等操作。

在时域（空域）中，锐化的方法不外乎是做掩码或做差分，同钝化的道理一样，无论是掩码还是差分都很难识别点之间的关联信息。下面的例子同样是在频域完成的，用传统的傅里叶分析方法（这里采用的是 DCT 变换）得到频域系数。

例 13-15，图像锐化。MATLAB 代码设置如下：

```
%读入 chess 信号
load chess;
%分别保存用 DCT 方法和小波方法的变换系数
blur1=X;
blur2=X;
%对原图像做二维离散余弦变换
ff1=dct2(X);
%对变换结果在频域做 BUTTERWORTH 滤波
for i=1:256
  for j=1:256
    ff1(i,j)=ff1(i,j)/(1+(32768/(i*i+j*j))^2);
  end
end
%重建变换后的图像
blur1=idct2(ff1);
%对图像做二层的二维小波分解
```

```
[c,l]=wavedec2(X,2,'db3');
csize=size(c);
%对低频系数进行放大处理，并抑制高频系数
for i=1:csize(2);
  if(abs(c(i))<300)
    c(i)=c(i)*2;
  else
    c(i)=c(i)/2;
  end
end
%通过处理后的小波系数重建图像
blur2=waverec2(c,l,'db3');
%显示 3 幅图像
subplot(221);image(wcodemat(X,192));colormap(gray(256));
title('原始图像','fontsize',18);
subplot(223);image(wcodemat(blur1,192));colormap(gray(256));
title('采用 DCT 方法锐化图像','fontsize',18);
subplot(224);image(wcodemat(blur2,192));colormap(gray(256));
title('采用小波方法锐化图像','fontsize',18);
```

得到的结果如图 13-46 所示。

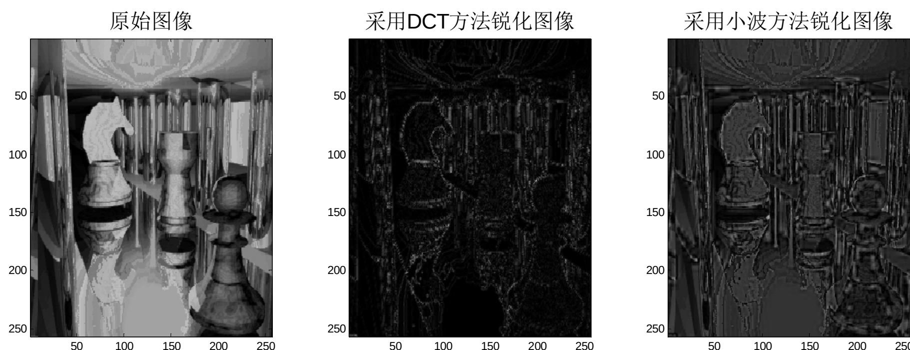


图 13-46　图像锐化

从结果可以看出，采用 DCT 方法进行高通滤波得到的高频结果比较纯粹，完全是原图像上的边缘信息，而在小波方法得到的结果中不止有高频成分，还有变换非常缓慢的低频成分，这是因为两者同样在小波系数上体现为绝对值较低的部分，但这些成分的存在对进行进一步分析并无多大影响。

最后比较这两个例子的时间复杂度。对 DCT 方法，需要做两次复杂度为 O($n$log$n$)的 DCT 变换，中间系数处理部分复杂度为 O(n)；而对小波变换，无论是分解、重构还是系数处理的复杂度都是 O(n)。

13.3.8　小波分析用于图像融合

图像融合是将同一图像的两个或更多的图像合成在一幅图像中，以便使合成后的图像比原来的任何一幅图像更容易为人们所理解。这一技术可应用于多频谱图像理解及医学图像处理等领域。在这些场合，同一物体部件的图像往往是采用不同的成像机理得到的。

例 13-16，下面用二维小波分析将上例中的 woman.mat 和 facets.mat 两幅图像融合在一

起。MATLAB 代码设置如下：

```
load woman;
X1=X;map1=map;
%画出原始图像
subplot(221);image(X1);colormap(map1);
title('woman');
axis square
load facets;
X2=X;map2=map;
for i=1:256
  for j=1:256
    if (X2(i,j)>100)
      X2(i,j)=1.2*X2(i,j);
    else
      X2(i,j)=0.5*X2(i,j);
    end
  end
end
subplot(222);image(X2);colormap(map2);
title('facets');
axis square
%用小波函数 sym4 对 X1 进行二层小波分解
[c1,s1]=wavedec2(X1,2,'sym4');
%对分解系数进行处理以突出轮廓部分，弱化细节部分
sizec1=size(c1);
for i=1：sizec1(2)
  c1(i)=1.2*c1(i);
end
%用小波函数 sym4 对 X2 进行二层小波分解
[c2,s2]=wavedec2(X2,2,'sym4');
%下面进行小波变换域的图像融合
c=c1+c2;
%减小图像亮度
c=0.5*c;
%对融合的系数进行重构
xx=waverec2(c,s1,'sym4');
%画出融合后的图像
subplot(223);image(xx);
title('融合图像');
axis square
```

输出结果如图 13-47 所示。

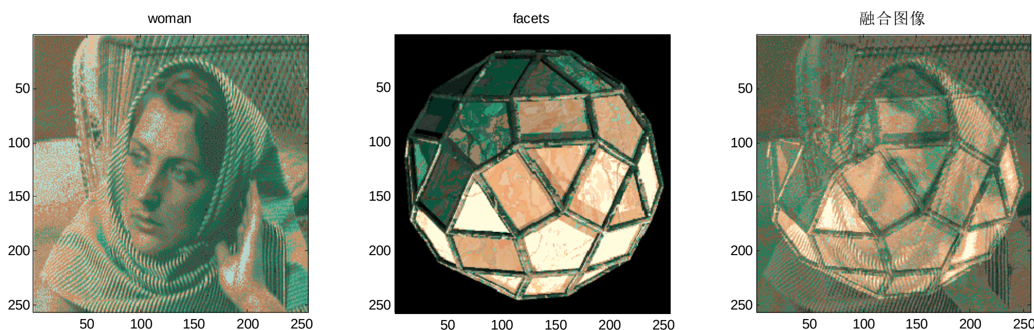


图 13-47　小波分析用于图像融合

13.3.9 小波分析用于图像分解

回顾从一维离散小波变换到二维的扩展，二维静态小波变换采用相似的方式，对行和列分别采用高通和低通滤波器。这样分解的结果仍然是 4 组图像：近似图像、水平细节图像、竖直细节图像和对角图像，与离散小波变换不同的只是静态小波分解得到的 4 幅图像与原图像尺寸一致，道理与一维情况相同。

二维离散小波变换同样只提供了一个函数 swt2，因为它不对分解系数进行下采样，所以单层分解和多层分解的结果是一样的，不需要另外提供多层分解的功能。

下面举一个使用 swt2 命令的例子，读者可以对比它与 dwt 处理结果的区别。

例 13-17，图像分解示例一。MATLAB 代码设置如下：

```
load woman
%使用 db1 小波对图像进行三层静态小波分解
[swa,swh,swv,swd]=swt2(X,3,'db1');
whos
colormap(map)
kp=0;
for i=1:3
subplot(3,4,kp+1),image(wcodemat(swa(:,:,i),192));
%显示第 i 层近似系数图像
title(['Approx,cfs,level',num2str(i)])
subplot(3,4,kp+2),image(wcodemat(swh(:,:,i),192));
title(['Horiz.Det.cfs level',num2str(i)])
subplot(3,4,kp+3),image(wcodemat(swv(:,:,i),192));
title(['Vert.Det.cfs level',num2str(i)])
subplot(3,4,kp+4),image(wcodemat(swd(:,:,i),192));
title(['Diag.Det.cfs level',num2str(i)])
kp=kp+4;
end
```

显示结果如图 13-48 所示，由于分解过程中没有改变信号的长度，所以在显示近似和细节系数时不需要重建。

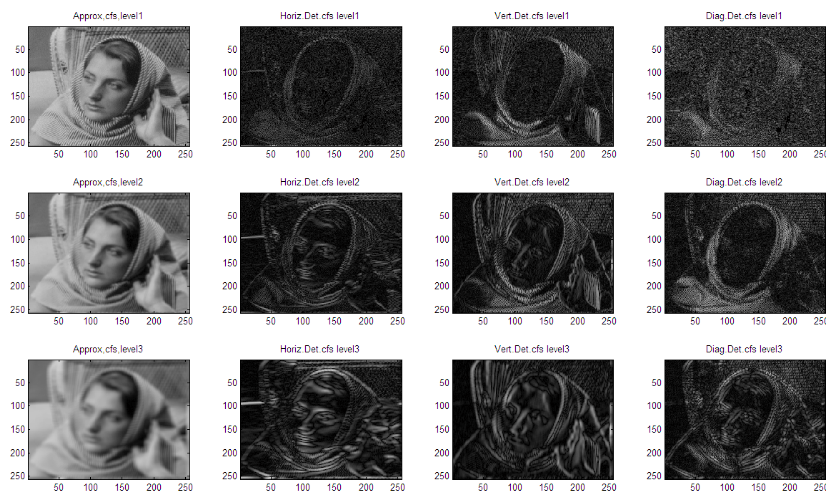


图 13-48 小波分析用于图像分解

同 idwt2 类似，MATLAB 对二维静态小波重建提供了 iswt2 函数；同 idwt 的去边情况类似，对经过重建滤波后的信号不做上采样（因为近似系数和细节系数大小都与原信号一致）。

同一维的静态小波重建一样，下面用例子说明如何将 iswt2 单纯用作滤波器来实现各层系数的重建。与一维情况不同的是，为了重建第 *j* 层近似系数，需要 4 次用到 iswt2 作为重建滤波器对第 *j*+1 层的系数进行滤波，在对某一个近似系数滤波的过程中，同样需要把其他 3 个系数指定为 0。

为了便于比较，本例接上面的二维静态分解的例子，直接利用对 woman 的分解结果，从中重建各级系数。

例 13-18，图像分解示例二。MATLAB 代码设置如下：

```
load woman
%使用 db1 小波对图像进行三层小波分解
[swa,swh,swv,swd]=swt2(X,3,'db1');
mzero=zeros(size(swd));
A=mzero;
%使用 iswt2 的滤波器功能，重建第 3 层的近似系数。为了避免 iswt 的合成运算，注意在重建过程中应保证其他各项系数为零
A(:,:,3)=iswt2(swa,mzero,mzero,mzero,'db1');
H=mzero;V=mzero;D=mzero;
for i=1;3
swcfs=mzero;swcfs(:,:,i)=swh(:,:,i);
H(:,:,i)=iswt2(mzero,swcfs,mzero,mzero,'db1');
swcfs=mzero;swcfs(:,:,i)=swv(:,:,i);
V(:,:,i)=iswt2(mzero,mzero,swcfs,mzero,'db1');
swcfs=mzero;swcfs(:,:,i)=swh(:,:,i);
H(:,:,i)=iswt2(mzero,mzero,mzero,swcfs,'db1');
end
%分别重建 1~3 级的各个细节系数，同样在重建某一系数的时候，要令其他系数为 0
A(:,:,2)=A(:,:,3)+H(:,:,3)+V(:,:,3)+D(:,:,3);
A(:,:,1)=A(:,:,2)+H(:,:,2)+V(:,:,2)+D(:,:,2);
%使用递推的方法建立第 1 层和第 2 层近似系数
colormap(map)
kp=0;
for i=1：3
subplot(3,4,kp+1),image(wcodemat(A(:,:,i),192));
title(['第',num2str(i),'层近似系数图像'])
subplot(3,4,kp+2),image(wcodemat(H(:,:,i),192));
title(['第',num2str(i),'层水平细节系数图像'])
subplot(3,4,kp+3),image(wcodemat(V(:,:,i),192));
title(['第',num2str(i),'层竖直细节系数图像'])
subplot(3,4,kp+4),image(wcodemat(D(:,:,i),192));
title(['第',num2str(i),'层对角细节系数图像'])
kp=kp+4;
end
%画出通过手工方法重建的各级小波系数图像
err=norm(A(:,:,2)-swa(:,:,2))
%求出用这种算法重建的第 2 层近似系数和分解系数之间的误差
```

误差结果：

```
err =
  9.4444e+004
```

显示的结果如图 13-49 所示。

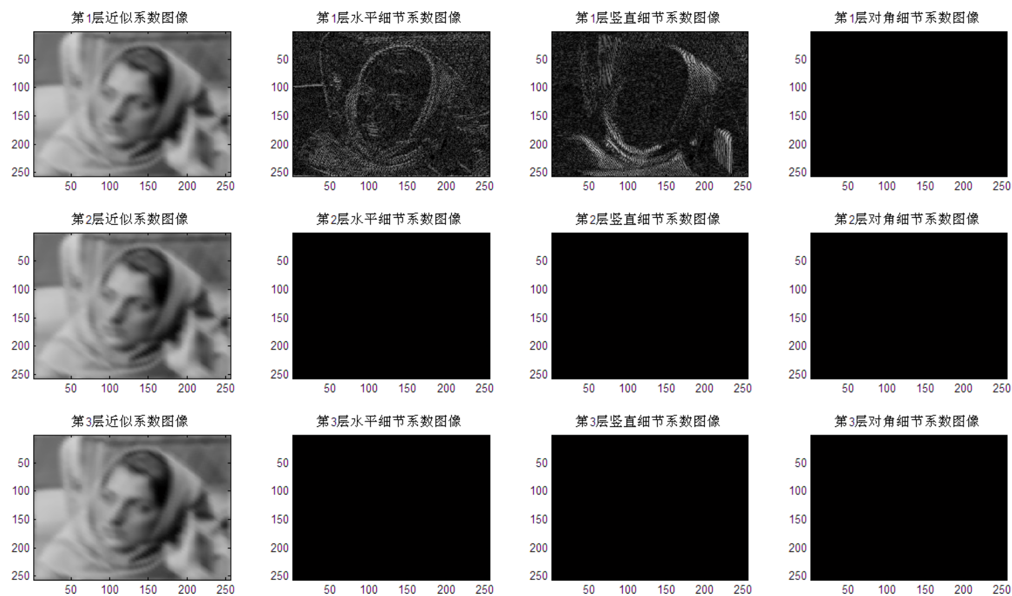


图 13-49　在 db1 小波下各级静态小波重建系数

13.4　本章小结

本章主要结合小波分析的基本概念和基本原理，详细讨论小波工具箱中一维连续（离散）小波分析工具的应用及小波在图像处理领域的应用，并结合 MATLAB 程序设计语言来说明其应用。读者通过本章的学习应能达到以下要求：

（1）了解小波分析的理论和方法。

（2）掌握小波工具箱的一些基本操作，如一维小波工具箱等。

（3）掌握小波分析和小波变换在图像处理中的应用。

第 14 章

Simulink 仿真基础

知识要点

Simulink 是 MATLAB 的重要组成部分，它提供建立系统模型、选择仿真参数和数值算法、启动仿真程序对该系统进行仿真、设置不同的输出方式来观察仿真结果等功能。Simulink 也是 MATLAB 最重要的组件之一，同时向用户提供一个动态系统建模、仿真和综合分析的集成环境。

在这个环境中，用户无须书写大量的程序，只需通过简单直观的鼠标操作，选取适当的库模块，即可构造出复杂的仿真模型。Simulink 的主要优点在于：适用面广、结构和流程清晰、仿真更为精细、模型内码更容易向 DPS/FPGA 等硬件移植。

学习要求	学习目标 知识点	了解	理解	应用	实践
	Simulink 的概念及其应用		√		
	Simulink 的工作环境				
	Simulink 模型创建		√		√
	过零检测和代数环	√			√



14.1 Simulink 概述

Simulink 是 MATLAB 系列工具软件包中最重要的组成部分。它能够对连续系统、离散系统及连续离散的混合系统进行充分的建模与仿真；能够借助其他工具直接从模型中生成可以直接投入运行的执行代码；可以仿真离散事件系统的动态行为；能够在众多专业工具箱的帮助下完成诸如 DSP、电力系统等专业系统的设计与仿真。

14.1.1 Simulink 的基本概念

Simulink 是一个进行动态系统建模、仿真和综合分析的集成软件包。它可以处理的系统包括：线性、非线性系统；离散、连续及混合系统；单任务、多任务离散事件系统。

在 Simulink 提供的图形用户界面（GUI）上，只要简单地进行鼠标的拖拉操作即可构造出复杂的仿真模型。它的外表以方块图形呈现，且采用分层结构。从建模角度讲，这既适于自上而下（Top-down）的设计流程（概念、功能、系统、子系统直至器件），又适于自下而上（Bottom-up）的逆程设计。

从分析研究角度讲，这种 Simulink 模型不仅能让用户知道具体环节的动态细节，而且能让用户清晰地了解各器件、各子系统、各系统间的信息交换，掌握各部分之间的交互影响。

在 Simulink 环境中，用户将摆脱理论演绎时需做理想化假设的无奈，观察到现实世界中摩擦、风阻、齿隙、饱和、死区等非线性因素和各种随机因素对系统行为的影响。在 Simulink 环境中，用户可以在仿真进程中改变感兴趣的参数，实时地观察系统行为的变化。

Simulink 的每个模块对于用户来说都相当于一个“黑匣子”，用户只需知道模块的输入和输出及模块功能即可，而不必管模块内部是怎么实现的。

因此，用户使用 Simulink 进行系统建模的任务就是如何选择合适的模块并把它们按照自己的模型结构连接起来，最后进行调试和仿真。如果仿真结果不满足要求，可以改变模块的相关参数再运行仿真，直到结果满足要求为止。至于在仿真时各个模块是如何执行的、各模块间是如何通信的、仿真的时间是如何采样的，以及事件是如何驱动的等细节问题，用户都不用去管，因为这些事情 Simulink 都解决了。如何添加和删除模块、如何连接各个模块，以及如何修改模块的参数和属性等问题在本章后面会陆续给予详细的介绍。

1．模块与模块框图

Simulink 模块框图是动态系统的图形显示，由一组称为模块的图标组成，模块之间的连接是连续的。每个模块代表了动态系统的某个单元，并且产生输出宏。模块之间的连线表明模块的输入端口与输出端口之间的信号连接。

模块的类型决定了模块输出与输入、状态和时间之间的关系；一个模块框图可以根据需要包含任意类型的模块。

模块代表了动态系统功能单元，每个模块包括输入、状态和输出等几个部分，模块的输出是仿真时间、输入或状态的函数。

模块中的状态是一组能够决定模块输出的变量，一般当前状态的值决定于以前时刻的状态值或输入，这样，具有状态变化的模块就必须存储以前时刻的输入或状态值，这样的模块称为记忆功能模块。

例如，Simulink 的积分（Integrator）模块就是典型的记忆功能模块。模块当前的输出是该模块从仿真开始到当前时刻这一时间段内输入信号的积分。

当前时刻的积分取决于历史输入，因此积分就是该模块的一组状态变量。另一个典型的例子就是 Simulink 中的单纯记忆（Memory）模块，该模块能够存储当前时刻的输入值，并在将来的某个时刻进行输出。

Simulink 的增益（Gain）模块是无状态变量的典型例子。增益模块的输出完全由当前的输入值决定，因此不存在状态变量。其他的无状态变量的模块还有求和模块（Sum）和点乘模块（Product）等。

Simulink 模块的基本特点是参数化，许多模块都具有独立的属性对话框，在对话框中可定义模块的各种参数，例如增益模块中的增益参数，这种调整甚至可以在仿真过程中实时进行，从而让用户能够找到最合适的参数值。这种能够在仿真运行过程中实时改变的参数又被称为可调参数，可以由用户在模块参数中任意指定。

Simulink 还允许用户创建自己的模块，这个过程又称为模块的定制。定制模块不同于 Simulink 中的标准模块，它可以由子系统封装得到，也可以采用 M 文件或 C 语言实现自己的功能算法，称为 S 函数。用户可以为定制模块设计属性对话框，并将定制模块合并到 Simulink 库中，使得定制模块的使用与标准模块的使用完全一样。

2．信号

Simulink 使用“信号”一词来表示模块的输出值。Simulink 允许用户定义信号的数据类型、数值类型（实数还是复数）和维数（一维数组还是二维数组）等。

Simulink 允许用户创建 Simulink 数据类型的实例（称为数据对象）来作为模块的参数和信号变量。

3．求解器

Simulink 模块指定了连续状态变量的时间导数，但本身没有定义这些导数的具体值，它们必须在仿真过程中通过微分方程的数值求解方法计算得到。Simulink 提供了一套高效、稳定、精确的微分方程数值求解方法（ODE），用户可以根据需要和模型特点选择适合的求解算法。

4．子系统

Simulink 允许用户在子系统的基础上构造更为复杂的模块，其中每一个子系统都是相对完整的、完成一定功能的模块框图。通过对子系统的封装，用户还可以实现带触发使用功能的特殊子系统。子系统的概念体现了分层建模的思想，是 Simulink 的重要特征之一。

5．零点穿越

在 Simulink 对动态系统进行仿真的过程中，一般在每一个仿真中都会检查系统状态变化的连续性。如果 Simulink 检测到了某个变量的不连续性，为了保持状态突变处系统仿真的准确性，仿真程序就会自动调整仿真步长，以适应这种变化。

动态系统中状态的突变对系统的动态特性具有重要的影响，例如弹性球在撞击地面时其速度方向会发生突变，这时，如果收集的时刻不是正好发生在仿真时刻当中（在相邻两步仿真之间），Simulink 的求解算法就不能正确反映系统的特性。

如果采用固定步长的算法，求解器就不能对此做相应的处理；相反，如果使用变步长的求解算法，Simulink 就会在确定突变时刻之后，在突变前增加额外的仿真计算，以保证突变前后计算的准确性。变步长的求解算法在状态变化缓慢时会增加仿真的步长，而在状态变化剧烈时则会减小仿真的步长。这样，在系统突变时刻，过小的仿真步长将会导致仿真时间的增加。

Simulink 采用一种称为零点穿越检测的方法来解决这个问题。

采用这种方法，模块首先记录下零点穿越的变量，每一个变量都是有可能发生突变的状态变量的函数。在突变发生时，零点穿越函数也从正数或负数穿过零点。通过观察零点穿越变量的符号变化，就可以判断仿真过程中系统状态是否发生了突变现象。

如果检测到穿越事件发生，Simulink 将通过对变量的以前时刻和当前时刻的插值来确定突变发生的具体时刻。然后，Simulink 调整仿真的步长，逐步逼近并跳过状态的不连续点，这样就避免了直接在不连续点上进行的仿真。

因为对于不连续系统而言，不连续点处的状态值可能是没有定义的。采用零点穿越检测技术，Simulink 可以准确地对不连续系统进行仿真。许多模块都支持这种技术，从而大大提高了系统仿真的速度和精度。

14.1.2　Simulink 的工作环境与启动

启动 Simulink 有如下三种方式：

（1）在 MATLAB 的命令行窗口中直接输入 simulink 命令。

（2）单击 MATLAB 主页选项中的按钮。

（3）单击 MATLAB 主页选项中的新建按钮→ Simulink Model 按钮。

打开的 Simulink 浏览器窗口如图 14-1 所示，单击 Blank Model 按钮，弹出如图 14-2 所示的窗口。

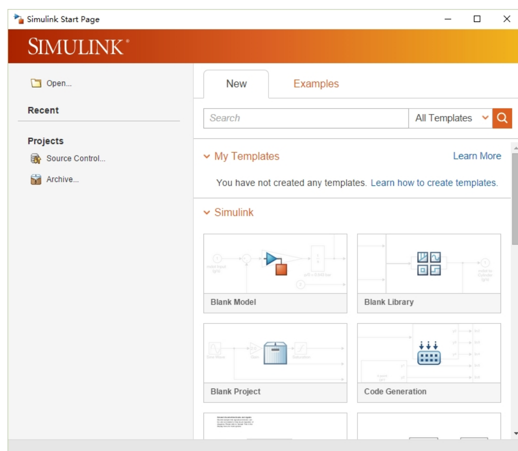


图 14-1　Simulink 模块库浏览器

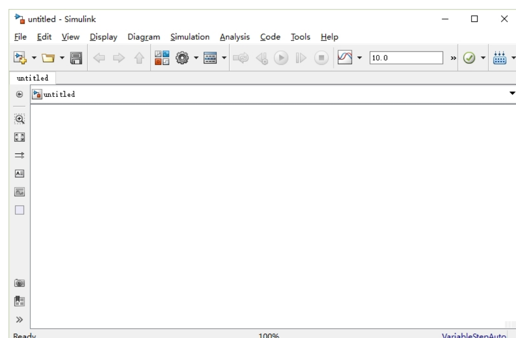


图 14-2　新建模型窗口

和 Windows 窗口类似，在 Simulink 的模块窗口和模块库窗口的 View 菜单下选择或取消选择 Toolbar 和 Status Bar 命令，就可以显示或隐藏工具条和状态条。在进行仿真的过程中，模型窗口的状态条会显示仿真状态、仿真进度和仿真时间等相关信息。

MATLAB 环境对话框可以让用户集中设置 MATLAB 及其工具软件包的使用环境，包括 Simulink 的环境设置。要在 Simulink 环境中打开该对话框，可以在 Simulink 窗口中选择 File →Simulink Preferences 菜单命令，出现如图 14-3 所示的对话框。

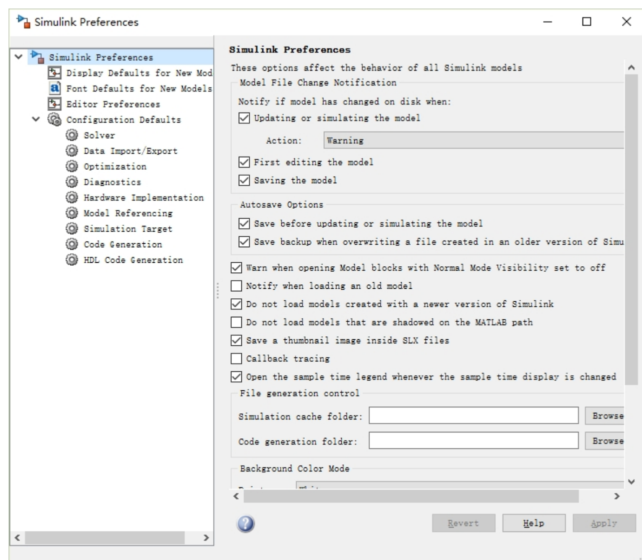


图 14-3　MATLAB 环境设置对话框

Simulink 主要由浏览器和模型窗口组成。前者为用户提供了展示 Simulink 标准模块库和专业工具箱的界面，而后者是用户创建模型方框图的主要地方。因此读者需要熟练掌握这些操作，知道所有菜单和按钮的功能。

14.1.3　Simulink 模型的特点

Simulink 的主要优点如下：

- 适应面广。可构造的系统包括：线性、非线性系统；离散、连续及混合系统；单任务、多任务离散事件系统。
- 结构和流程清晰。它外表以方块图形式呈现，采用分层结构，既适于自上而下的设计流程，又适于自下而上的逆程设计。
- 仿真更为精细。它提供的许多模块更接近实际，为用户摆脱理想化假设的无奈开辟了途径。
- 模型内码更容易向 DPS、FPGA 等硬件移植。

而使用 Simulink 建立的模型具有以下 3 个特点：

- 仿真结果的可视化。

- 模型的层次性。
- 可封装子系统。

下面通过一个 Simulink 提供的演示示例来说明上述特点。

（1）在 Simulink 模型窗口中，通过菜单命令 Help→Simulink→Simulink Help 打开如图 14-4 所示的帮助窗口。

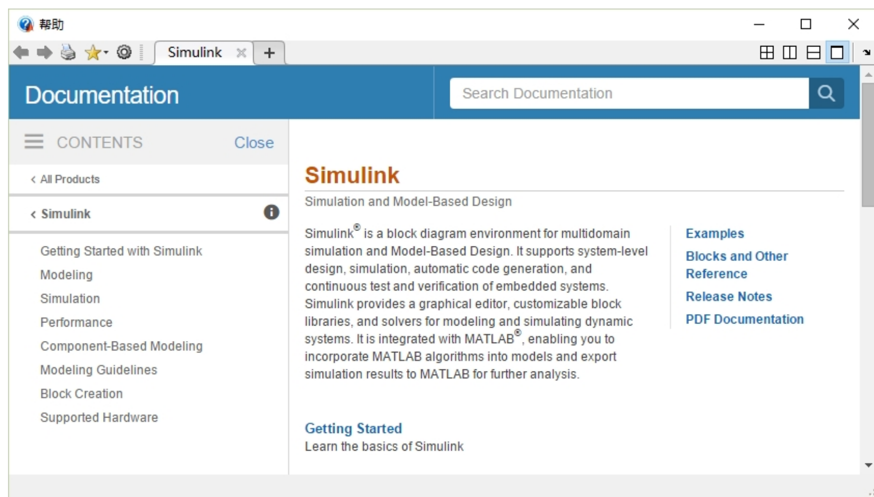


图 14-4　MATLAB R2018a 帮助窗口

（2）在 Examples 选项卡中选择 Simulink→General Applications→Thermal Model of a House 选项，单击文本框右边的 Open This Example 按钮，打开如图 14-5 所示的窗口。

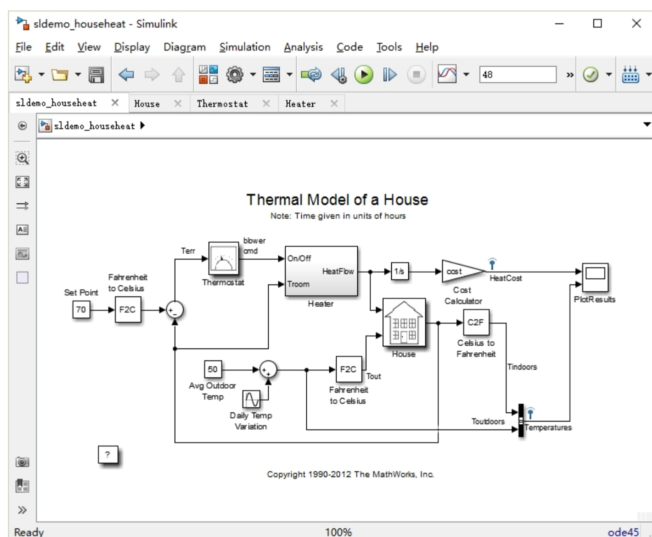


图 14-5　thermo 演示模型

对于上述特点，读者在学完后续章节后将会有更加深刻的理解。

（3）单击 Run 按钮，可以看到如图 14-6 所示的仿真结果。

（4）双击模型图标中的 House 模块，可以弹出如图 14-7 所示的 House 子系统图标。

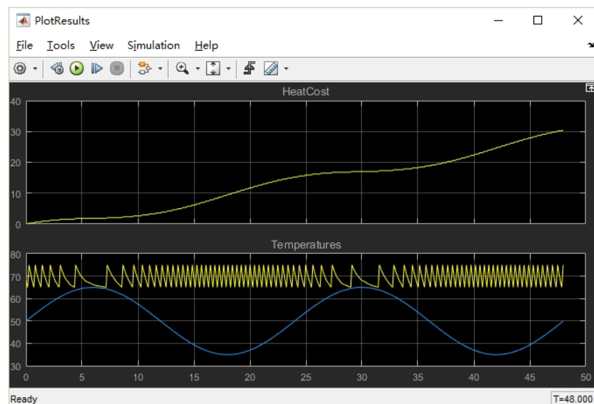


图 14-6　仿真结果可视化

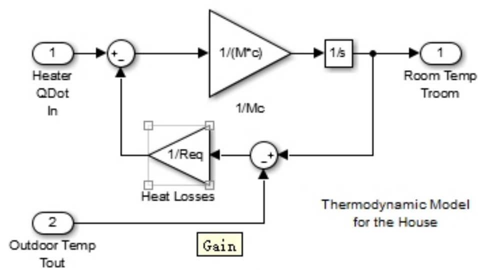


图 14-7　House 子系统图标

14.1.4　Simulink 模块的组成

1．应用工具

Simulink 软件包的一个重要特点是它完全建立在 MATLAB 的基础上。因此，MATLAB 各种丰富的应用工具箱也可以完全应用到 Simulink 环境中来，这无疑大大扩展了 Simulink 的建模和分析能力。

基于 MATLAB 的所有工具箱都是经过全世界各个领域的专家和学者共同研究的最新成果，每一个工具箱都可谓是千锤百炼，其领域涵盖了自动控制、信号处理和系统辨识等十多个学科。并且随着科学技术的发展，MATLAB 的应用工具箱始终处在不断发展完善之中。

MATLAB 应用工具箱的另一个特点是完全的开放性，任何用户都可随意浏览、修改相关的 M 文件，创建满足用户特殊要求的工具箱。由于其中的算法有很多是相当成熟的产品，用户可以采用 MATLAB 自带的编译器将其编译成可执行代码，并嵌入到硬件当中直接执行。

2．实时工作室

Simulink 软件包中的实时工作室（Real-Time Workshop）可以将 Simulink 的仿真框图直接转换成 C 语言代码，从而直接从系统仿真过渡到系统实现。该工具支持连续、离散和连续-离散混合系统。用户完成 C 语言代码的转换后，可以直接进行汇编，生成可执行文件。

实时工作室的优点如下：

- 无须手工编写代码和复杂的调试过程，就可以完成从动态系统设计到最后代码实现的全过程，包括控制算法设计、信号处理、研究动态系统等，都可以借助 Simulink 的可视化方框图进行方便的设计。
- 一旦 Simulink 完成了系统的设计，用户可以借助工具生成嵌入式的代码，进行编译、连接之后，直接嵌入到硬件设备中。
- 异步仿真。由于 MATLAB 中可以以 ASCII 码或二进制文件记录仿真所经历的时间，用户可以将仿真过程放在客户机或发送到远程计算机中进行仿真。
- Simulink 的实时工作室支持大量不同的系统和硬件设备，并且具有友好的图形用户界面，使用起来更加方便、灵活。

3．状态流模块

MATLAB 使用的是 stateflow，Simulink 的模块窗中包含了 stateflow 的模块，用户可以在该模块中设计基于状态变化的离散事件系统。将该模块放入 Simulink 模型当中，就可以创建包含离散事件子系统的更为复杂的模型。

4．扩展的模块集

如同众多的应用工具箱扩展了 MATLAB 的应用范围一样，MathWorks 公司为 Simulink 提供了各种专门的模块集（BlockSet）来扩展 Simulink 的建模和仿真能力。这些模块集涉及电力、非线性控制、DSP 系统等不同领域，满足了 Simulink 对不同系统仿真的需要。这些模块集包括：

- CDMA Reference Blockset：用于满足 IS—95A 无线通信标准系统的设计与仿真的模块库。
- Comunications Blockset：满足对通信系统物理层建模与仿真需要的模块库。
- Dials＆Gauges Blockset：用于监控 Simulink 信号与模块参数的图形工艺。
- DSP Blockset：用于数字信号处理系统仿真和设计的模块库。
- Fix-point Blockset：用于固定点系统仿真、设计和代码生成的模块库。
- Nonlinear Control Desigo(NCD) Blockset：为非线性系统的设计提供时域优化方法，它可以基于指定的时域性能约束自动调整系统参数。
- Power system Blockset：用于电力系统设计和仿真的模块库。

14.1.5　Simulink 中的数据类型

在计算机编程语言中，数据类型决定了分配给一个数据的存储资源，决定了数据表示的精度、动态范围、性能和存储资源。MATLAB 语言也是一种编程语言，因此 Simulink 也允许用户说明 Simulink 模型中信号和模块参数的数据类型。

Simulink 在开始仿真之前及仿真过程中会进行一个额外的检查（系统自动进行，无须手动设置），以确认模型的类型安全性。所谓模型的类型安全性，是指保证该模型产生的代码不会出现上溢或下溢，不至于产生不精确的运行结果。使用 Simulink 默认数据类（double）的模型都是安全的固有类型。

1．Simulink 支持的数据类型

Simulink 支持所有的 MATLAB 内置数据类型，内置数据类型是指 MATLAB 自己定义的数据类型。表 14-1 列出了所有的 MATLAB 内置数据类型。

表 14-1　MATLAB 内置数据类型

数据类型	类型说明	数据类型	类型说明
double	双精度浮点类型	int16	有符号 16 位整数
single	单精度浮点类型	uint16	无符号 16 位整数
int8	有符号 8 位整数	int32	有符号 32 位整数
uint8	无符号 8 位整数	uint32	无符号 32 位整数

除了内置数据类型，Simulink 还定义了布尔类型，取值为 0 和 1，它们内部的表示是 uint8（无符号 8 位整数）。所有的 Simulink 模块都默认为 double 类型的数据，但有些模块需要布尔类型的输入，而另外一些模块支持多数据类型输入，还有一些支持复数信号。

关于模块的输入/输出信号支持的数据类型的详细说明，用户可以通过模块参数设置对话框（双击模块图标就会弹出对话框）来查看。如果在一个模块的参数设置对话框中没有说明它支持的数据类型的选项，那就表示它只支持 double 类型的数据。

选择模型窗口的 Display→Signal & Ports→Port Data Types 选项可以查看信号的数据类型和模块输入/输出端口的数据类型，如图 14-8 所示。

在设置模块参数时，指定一个值为某一数据类型的方法为 type(value)。例如要把常数模块的参数设为 1.0 单精度表示，则可以在参数对话框中输入 single(1.0)。如果模块不支持所设置的数据类型，就会弹出错误警告。

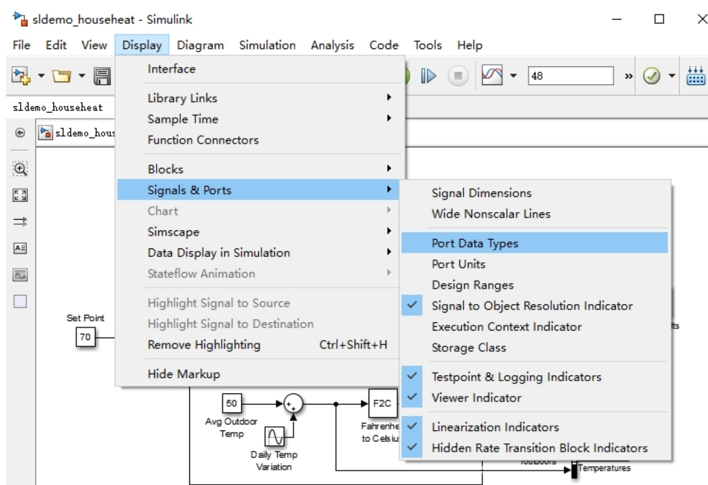


图 14-8　查看信号的数据类型

2．数据类型的传播

构造模型时会将各种不同类型的模块连接起来，而这些不同类型的模块所支持的数据类型往往不完全相同。如果用直线连接在一起的两个模块所支持的数据类型（注意是指输出/输入信号的数据类型，而不是模块参数的数据类型）有冲突，那么当进行仿真、查看端口数据类型或是更新数据类型时就会弹出一个提示对话框，告诉用户出现冲突的信号和端口，而且有冲突的信号的路径会被加亮显示。这时就可以通过在有冲突的模块之间插入一个 Data Type Conversion 模块来解决类型冲突。

一个模块的输出一般是模块输入和模块参数的函数。而在实际建模过程中，输入信号的数据类型和模块参数的数据类型往往是不同的，Simulink 在计算这种输出时会把参数类型转换为信号的数据类型。当信号的数据类型无法表示参数值时，Simulink 将中断仿真，并给出错误信息。

如图 14-9 所示的示例模型，常数模块的输出信号类型设置为布尔型，而连续信号积分器只接收 double 类型信号，所以弹出出错提示框。这时可以在示例模型中插入一个 Data Type

Conversion 模块，并将其输出改成 double 数据类型就不会报错了，如图 14-10 所示。

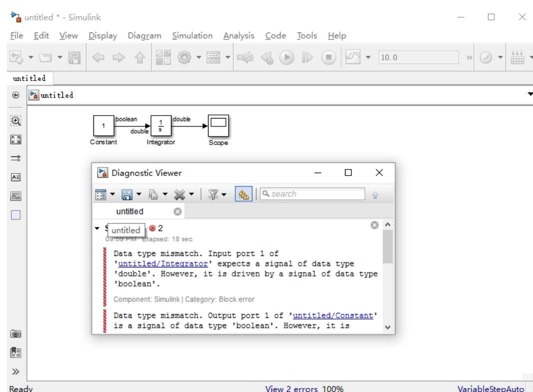


图 14-9　数据类型示例模型

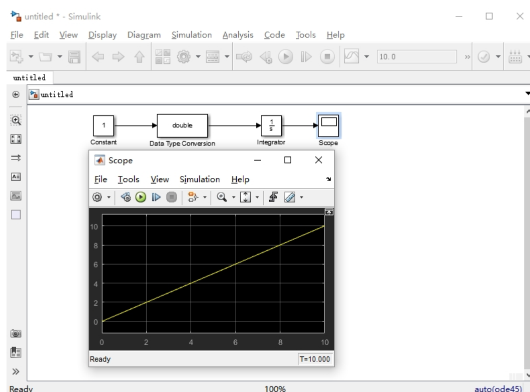


图 14-10　修改后的示例

如果信号的数据类型能够表示参数的值仅仅是损失表示的精度，那么 Simulink 会继续仿真，并在 MATLAB 命令行窗口中给出一个警告信息。

在图 14-11 中，常数模块的常数值参数设置为 uint8(1)，输出数据类型设为 Inherit from Constant value；增益模块的增益参数设置为 double(3.2)，输出数据类型设置为 Same as input，参数数据类型设置为 Interit via internal rule。Simulink 会把 3.2 的整数部分截取，并转换成无符号数，所以最后结果为 3。

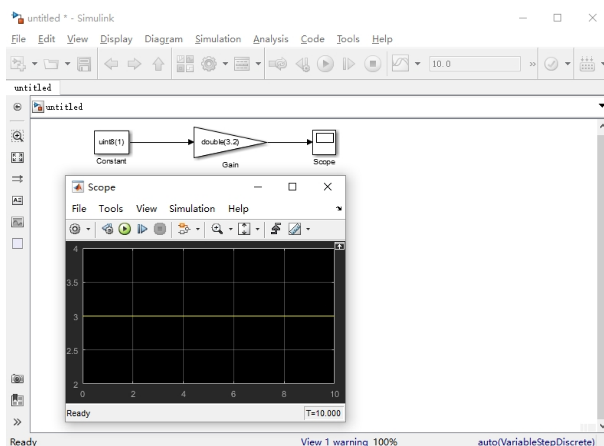


图 14-11　数据精度损失的示例

3．使用复数信号

Simulink 中默认的信号值都是实数，但在实际问题中有时需要处理复数信号。在 Simulink 中通常用下面两种方法来建立处理复数信号的模型，如图 14-12 所示。

向模型中加入一个 Constant 模块，将其参数设为复数。分别生成复数的虚部和实部，再用 Real-Image to Complex 模块把它们联合成一个复数；或分别生成复数的幅值和幅角，再用 Magnitue-Angle to Complex 模块把它们联合成一个复数。

同样可以在 Simulink 中利用相关的模块将一个复数分解成虚部和实部或是幅值和幅角，而且 Simulink 中有许多模块都接收复数输入并对复数进行运算，请读者自己查阅帮助文件来

了解这些模块。

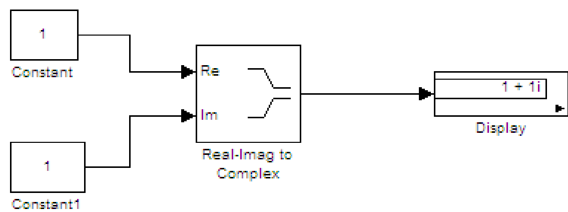


图 14-12　引入复数

14.1.6　Simulink 中的模块和模块库

用 Simulink 建模的过程可以简单地理解为从模块库中选择合适的模块，然后将它们连接在一起，最后进行调试仿真。

模块库的作用就是提供各种基本模块，并将它们按应用领域及功能进行分类管理，以方便用户查找。库浏览器将各种模块库按树状结构进行罗列，以便用户快速地查询所需模块，同时它还提供了按名称查找的功能。库浏览器中模块的多少取决于用户安装的多少，但至少应该有 Simulink 库，用户还可以自定义库。

模块是 Simulink 建模的基本元素，了解各个模块的作用是熟练掌握 Simulink 的基础。库中各个模块的功能可以在库浏览器中查到。下面详细介绍 Simulink 库中几个常用子库中的常用模块的功能，如表 14-2～表 14-11 所示。

表 14-2　Commonly Used Blocks 子库

模 块 名	功　　能
Bus Creator	将输入信号合并成向量信号
Bus Selector	将输入向量分解成多个信号，输入只接收 Mux 和 Bus
Creator	输出的信号
Constant	输出常量信号
Data Type Conversion	数据类型的转换
Demux	将输入向量转换成标量或更小的标量
Discrete-Time Integrator	离散积分器模块
Gain	增益模块
In1	输入模块
Integrator	连续积分器模块
Logical Operator	逻辑运算模块
Mux	将输入的向量、标量或矩阵信号合成
Out1	输出模块
Product	乘法器，执行标量、向量或矩阵的乘法
Relational Operator	关系运算，输出布尔类型数据

（续表）

模 块 名	功　　能
Saturation	定义输入信号的最大值和最小值
Scope	输出示波器
Subsystem	创建子系统
Sum	加法器
Switch	选择器，根据第二个输入信号来选择输出第一个还是第三个信号
Terminator	终止输出，用于防止模型最后的输出端没有接收任何模块时报错
Unit Delay	单位时间延迟

表 14-3　Continuous 子库

模 块 名	功　　能
Derivative	数值微分
Integrator	积分器与 Commonly Used Blocks 子库中的同名模块一样
State-Space	创建状态空间模型 dx/dt = $Ax + Bu$ $y = Cx + Du$
Transport Delay	定义传输延迟。如果将延迟设置得比仿真步长大，可以得到更精确的结果
Transfer Fcn	用矩阵形式描述的传输函数
Variable Transport Delay	定义传输延迟，第一个输入为接收输入，第二个输入接收延迟时间
Zero-Pole	用矩阵描述系统零点，用向量描述系统极点和增益

表 14-4　Discontinuities 子库

模 块 名	功　　能
Coulomb&Viscous Friction	刻画在零点的不连续性，y = sign(x) × (Gain × abs(x) + Offset)
Dead Zone	产生死区，当输入在某一范围取值时输出为 0
Dead Zone Dynamic	产生死区，当输入在某一范围取值时输出为 0。与 Dead Zone 不同的是，它的死区范围在仿真过程中是可变的
Hit Crossing	检测输入是上升经过某一值还是下降经过这一值或是固定在某一值，用于过零检测
Quantizer	按相同的间隔离散输入
Rate Limiter	限制输入的上升和下降速率在某一范围
Rate Limiter Dynamic	限制输入的上升和下降速率在某一范围。与 Rate Limiter 不同的是，它的范围在仿真过程中是可变的
Relay	判断输入与某两阈值的大小关系。当大于开启阈值时，输出为 on；当小于关闭阈值时，输出为 off；当在两者之间时，输出不变
Saturation	限制输入在最大和最小范围之内
Saturation Dynamic	限制输入在最大和最小范围之内。与 Saturation 不同的是，它的范围在仿真过程中是可变的
Wrap To Zero	当输入大于某一值时输出 0，否则输出等于输入

表 14-5　Discrete 子库

模 块 名	功　　能
Difference	离散差分，输出当前值减去前一时刻的值
Discrete Derivative	离散偏微分
Discrete State-Space	创建离散状态空间模型 $x(n+1) = Ax(n) + Bu(n)$和 $y(n) = Cx(n) + Du(n)$
Discrete Filter	离散滤波器
Discrete Transfer Fcn	离散传输函数
Discrete Zero-Pole	离散零极点
Discrete-Time Integrator	离散积分器
First-Order Hold	一阶保持
Integer Delay	整数倍采样周期的延迟
Memory	存储单元，当前输出是前一时刻的输入
Transfer Fcn First Order	一阶传输函数，单位的直流增益
Zero-Order Hold	零阶保持

表 14-6　Logic and Bit Operations 子库

模 块 名	功　　能
Bit Clear	将向量信号中某一位置设置为 0
Bit Set	将向量信号中某一位置设置为 1
Bitwise Operator	对输入信号进行自定义的逻辑运算
Combinatorial Logic	组合逻辑，实现一个真值表
Compare To Constant	定义如何与常数进行比较
Compare To Zero	定义如何与零进行比较
Detect Change	检测输入的变化，如果输入的当前值与前一时刻的值不等，则输出 TRUE，否则输出 FALSE
Detect Decrease	检测输入是否下降，是则输出 TRUE，否则输出 FALSE
Detect Fall Negative	若输入当前值是负数，前一时刻值为非负，则输出 TRUE，否则输出 FALSE
Detect Fall Nonpositive	若输入当前值是非正，前一时刻值为正数，则输出 TRUE，否则输出 FALSE
Detect Increase	检测输入是否上升，是则输出 TRUE，否则输出 FALSE
Detect Rise Nonnegative	若输入当前值是非负，前一时刻值为负数，则输出 TRUE，否则输出 FALSE
Detect Rise Positive	若输入当前值是正数，前一时刻值为非正，则输出 TRUE，否则输出 FALSE
Extract Bits	从输入中提取某几位输出
Interval Test	检测输入是否在某两个值之间，是则输出 TRUE，否则输出 FALSE
Logical Operator	逻辑运算
Relational Operator	关系运算
Shift Arithmetic	算术平移

表 14-7　Math Operations 子库

模 块 名	功　　能
Abs	求绝对值
Add	加法运算
Algebraic Constraint	将输入约束为零，主要用于代数等式的建模
Assignment	选择输出输入的某些值
Bias	将输入加一个偏移，$Y= U+$ Bias
Complex to Magnitude-Angle	将输入的复数转换成幅度和幅角
Complex to Real-Imag	将输入的复数转换成实部和虚部
Divide	实现除法或乘法
Dot Product	点乘
Gain	增益，实现点乘或普通乘法
Magnitude-Angle to Complex	将输入的幅度和幅角合成复数
Math Function	实现数学函数运算
Matrix Concatenation	实现矩阵的串联
MinMax	将输入的最小值或最大值输出
Polynomial	多项式求值，多项式的系数以数组的形式定义
MinMax Running Resettable	将输入的最小值或最大值输出，当有重置信号 R 输入时，输出被重置为初始值
Product of Elements	将所有输入实现连乘
Real-Imag to Complex	将输入的两个数当成一个复数的实部和虚部合成一个复数
Reshape	改变输入信号的维数
Rounding Function	将输入的整数部分输出
Sign	判断输入的符号，若为正输出 1，为负输出-1，为零输出 0
Sine Wave Function	产生一个正弦函数
Slider Gain	可变增益
Subtract	实现加法或减法
Sum	加法或减法
Sum of Elements	实现输入信号所有元素的和
Trigonometric Function	实现三角函数和双曲线函数
Unary Minus	一元的求负
Weighted Sample Time Math	根据采样时间实现输入的加法、减法、乘法和除法，只对离散信号适用

表 14-8　Ports & Subsystems 子库

模 块 名	功　　能
Configurable Subsystem	用于配置用户自建模型库，只在库文件中才可用
Enable	使能模块，只能用在子系统模块中
Enabled and Triggered Subsystem	包括使能和边沿触发模块的子系统模板

（续表）

模 块 名	功　能
Enabled Subsystem	包括使能模块的子系统模板
For Iterator Subsystem	循环子系统模板
Function-Call Generator	实现循环运算模板
Function-Call Subsystem	包括输入/输出和函数调用触发模块的子系统模板
If	条件执行子系统模板，只在子系统模块中可用
If Action Subsystem	由 If 模块触发的子系统模板
Model	定义模型名字的模块
Subsystem	只包括输入/输出模块的子系统模板
Subsystem Examples	子系统演示模块，在模型中双击该模块图标可以看到多个子系统示例
Switch Case	条件选择模块
Switch Case Action Subsystem	由 Switch Case 模块触发的子系统模板
Trigger	触发模块，只在子系统模块中可用
Triggered Subsystem	触发子系统模板
While Iterator Subsystem	条件循环子系统模板

表 14-9　Sinks 子库

模 块 名	功　能
Display	显示输入数值的模块
Floating Scope	浮置示波器，由用户来设置所要显示的数据
StopSimulation	当输入不为零时，停止仿真
To Workspace	将输入和时间写入 MATLAB 工作区的数组或结构中
To File	将输入和时间写入 MAT 文件
XY Graph	将输入分别当成 x、y 轴数据绘制成二维图形

表 14-10　Sources 子库

模 块 名	功　能
Band-Limited White Noise	有限带宽的白噪声
Chirp Signal	产生 Chirp 信号
Clock	输出当前仿真时间
Constant	输出常数
Counter Free-Running	自动计数器，发生溢出后又从 0 开始
Counter Limited	有限计数器，当计数到某一值后又从 0 开始
Digital Clock	以数字形式显示当前的仿真时间
From File	从 MAT 文件中读取数据
From Workspace	从 MATLAB 工作区读取数据
Pulse Generator	产生脉冲信号

（续表）

模 块 名	功　　能
Ramp	产生按某一斜率的数据
Random Number	产生随机数
Repeating Sequence	重复输出某一数据序列
Signal Builder	具有 GUI 界面的信号生成器，在模型中双击模块图标可看到图形用户界面，利用该界面可以直观地构造各种信号
Signal Generator	信号产生器
Sine Wave	产生正弦信号
Step	产生阶跃信号
Uniform Random Number	按某一分布在某一范围生成随机数

表 14-11　User-Defined Functions 子库

模 块 名	功　　能
Fcn	简单的 MATLAB 函数表达式模块
Embedded MATLAB Function	内置 MATLAB R2018a 函数模块，在模型窗口中双击该模块图标就会弹出 M 文件编辑器
M-file SFunction	用户使用 MATLAB 语言编写的 S 函数模块
MATLAB Fcn	对输入进行简单的 MATLAB 函数运算
SFunction	用户按照 S 函数的规则自定义的模块，可以使用多种语言进行编写
SFunction Builder	具有 GUI 界面的 S 函数编辑器，在模型中双击该模块图标可看到图形用户界面，利用该界面可以方便地编辑 S 函数模块
SFunction Examples	S 函数演示模块，在模型中双击该模块图标可以看到多个 S 函数示例

14.1.7　Simulink 常用工具

1．仿真加速器

Simulink 加速器能够提高模型仿真的速度，它的基本工作原理是利用 Real-Time Workshop 工具将模型方框图转换成 C 语言代码，然后采用编译器将 C 语言代码编译成可执行代码。由于用可执行代码取代了原有的 MATLAB 解释器，仿真的速度可能会有本质的提高。

Simulink 既可以工作在正常模式（Norma1）下，也可以工作在加速模式（Accelerator）下。工作在加速模式下时，Simulink 将 C 语言代码编译成 MEX 文件，在编译过程中，对原有的 C 语言代码进行优化重组，可以极大地提高模型仿真的速度。并且模型越复杂，提高的程度越明显，一般来说，可以将仿真的速度提高 2～6 倍。

为了使 Simulink 工作在加速模式下，在模型窗口中选择 Simulation→Accelerator 命令。如果要恢复到正常模式，可以选择选择 Simulation→Normal 命令。

值得注意的是，加速器产生的 C 语言代码只能作为加速仿真之用，如果需要生成其他的作用 C 语言代码（例如产生嵌入式 C 语言代码）就必须使用 Real-timeworkshop 工具。

如果用户改变了模型的结构，例如增加或删除模型中的模块，C 语言代码将重新进行编译。下面的操作将会影响模型的结构。

- 改变积分的算法。
- 增减模块或改变模块之间的连接。
- 改变模块输入/输出端口的数目。
- 改变模型当中状态变量的数目。
- 改变 Trigonometric Function 模块中的函数。
- 改变 Sum 模块中所用的符号。

在模型仿真过程中，如果用户对模型所做的修改影响到模型的结构，Simulink 将忽略这种改变，并且出现警告信息。用户如果要想使修改生效，必须停止仿真过程，待修改完成后，重新开始。当然，简单的修改，如改变 Gain 模块的增益值，不会产生警告消息，这表示加速器认可这种修改。

需要注意的是，加速器不会显示仿真过程本身产生的警告消息，如被零除和数据溢出，这点与前面所讲的情况不大一样。

2．模型比较工具

Simulink 中的模型比较工具（Model Differencing Tool）可以让用户迅速找到两个模型之间的不同点，例如两个相同模型之间的不同版本。

模型比较工具的界面分成 3 个子窗口：左上方的子窗口显示的是第一个模型的具体内容；右上方的子窗口显示的是第二个模型的具体内容；下方的子窗口显示的是两个模型当中相同模块的不同参数。

3．仿真统计表

Simulink 中的仿真统计表生成器可以根据仿真过程中的数据生成一个称为仿真统计表（Simulation Profiler）的报告。该报告可以显示该模型在仿真过程中每一个功能模块所花费的时间，从而可以让用户确定决定模型仿真速度的主要因素，为进一步优化仿真模型提供帮助。

为了使用该工具，用户需首先打开指定的模型，选择 Simulink→Profiler 选项，然后启动仿真。当仿真结束后，Simulink 将生成并在 MATLAB 帮助浏览器中显示仿真的统计表。最后，Simulink 会将该统计表保存到 MATLAB 的工作目录下。

14.1.8　Simulink 的实例演示

下面先向读者介绍一个非常简单的示例，旨在使读者在进行深入学习之前先对 Simulink 有一个感性的认识。

数学模型：$x=\cos t, y=\int_0^t x(t)\mathrm{d}t$。

目的：动态画圆，并显示结果的波形。

[1] 单击 MATLAB 工具条上的新建→Simulink Model 按钮，弹出如图 14-13 所示的窗口。

[2] 单击 Library Browse 按钮。选中 Sources 库，然后在右边选中 Sine Wave 模块并

按住鼠标左键不放，将它拖到如图 14-13 所示的窗口中。然后选中 Commonly Used Blocks 库，选中 Integrator 模块并按住鼠标左键不放，拖到如图 14-13 所示的窗口中；按同样的方法添加 Sinks 库中的 XY Graph 模块。

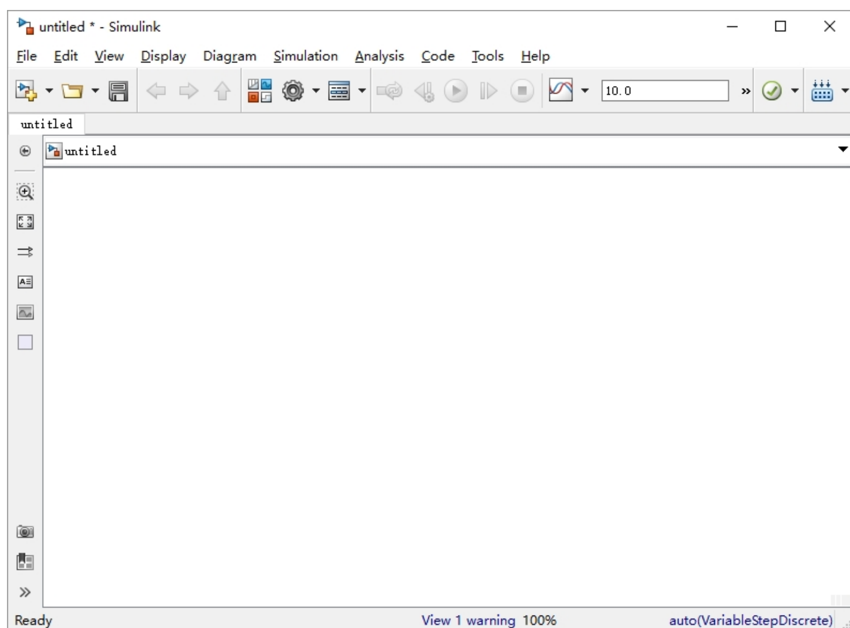


图 14-13　Simulink 模型创建窗口

[3] 连接模块，如图 14-14 所示。连接模块的操作方法是将光标指向源模块的输出端口，当光标变成十字架形时按住鼠标左键不放，然后拖动鼠标到目标模块输入窗口后松开。

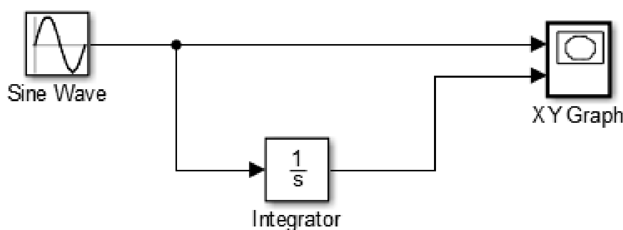


图 14-14　简单示例模型

[4] 设置 Sine Wave 模块的参数。双击 Sine Wave 模块，弹出如图 14-15 所示的参数设置对话框，设置 Phase（相位）为 3.14/2，然后单击 OK 按钮。

[5] 设置 Integrator 模块的参数。双击 Integrator 模块，弹出如图 14-16 所示的参数设置对话框，设置 Initial condition（初始值）为 0，然后单击 OK 按钮。

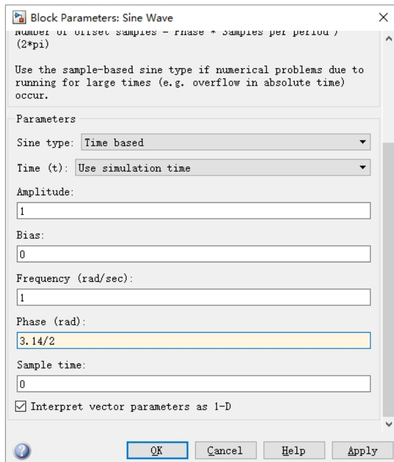


图 14-15　Sine Wave 参数设置

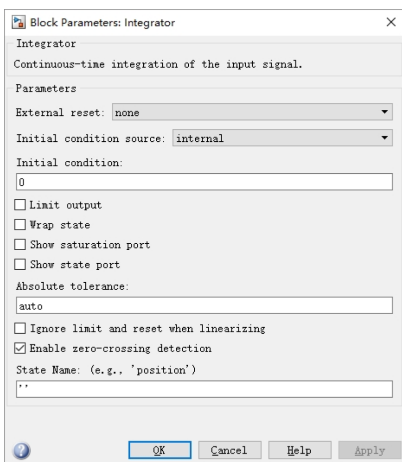


图 14-16　Integrator 参数设置

[6] 设置 XY Graph 模块的参数。双击 XY Graph 模块，弹出如图 14-17 所示的参数设置对话框，设置 x 的范围为-1.5～1.5，y 的范围为-1.2～1.2，然后单击 OK 按钮。

[7] 接下来运行仿真，弹出如图 14-18 所示的图形。

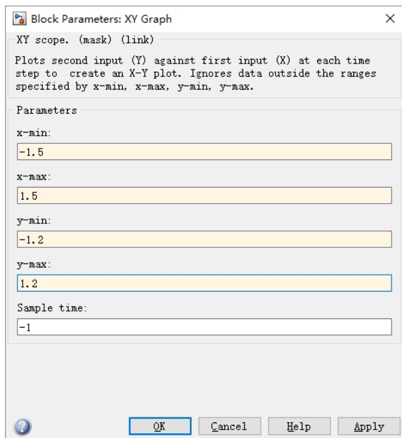


图 14-17　XY Graph 参数设置

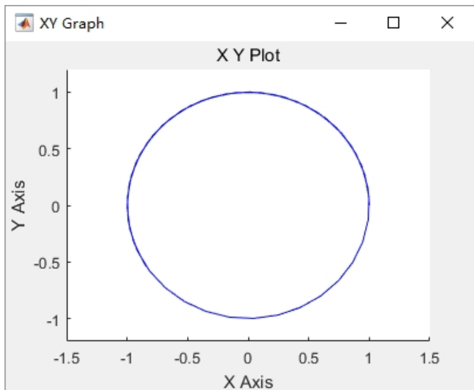


图 14-18　输出圆

14.2　Simulink 模型创建

模块是建立 Simulink 模型的基本单元。利用 Simulink 进行系统建模就是用适当的方式把各种模块连接在一起。创建模型方框图是 Simulink 进行动态系统仿真的第一步。Simulink 为用户创建系统仿真模型提供了友好的可视化环境，用户通过鼠标的单击和拖动操作就能完成创建模型的大部分工作。

在 Simulink 环境中，系统模型是由方框图表示的，模块和信号线是方框图的基本组成单位。因此，了解模块与信号线的概念和使用是创建模型的第一步。

另外，新版本 MATLAB 中的 Simulink 新增了许多功能，在随后几节中将逐一进行介绍。

本节主要介绍 Simulink 创建模型中的有关概念、相关的工具和操作方法，旨在使读者熟悉 Simulink 环境的使用和模型创建的基本操作，为以后进一步深入学习打下基础。

14.2.1 Simulink 模块的基本操作

Simulink 的模块库提供了大量模块。单击模块库浏览器中 Simulink 前面的+号，将看到 Simulink 模块库中包含的子模块库，单击所需要的子模块库，在右边的窗格中将看到相应的基本模块，选择所需的基本模块，用鼠标将其拖到模型编辑窗口。

同样，在模块库浏览器左侧的 Simulink 栏上单击鼠标右键，在弹出的快捷菜单中选择 Open the 'Simulink' Libaray 命令，将打开 Simulink 基本模块库窗口。单击其中的子模块库图标，打开子模块库，找到仿真所需要的基本模块。

表 14-12 和表 14-13 汇总了 Simulink 中对模块、直线和信号标签进行各种操作的方法，其中有许多操作并不是唯一的，读者自己实践一下就可以摸索出其他的操作方法。

表 14-12 对模块进行操作

任　务	Microsoft Windows 环境下的操作
选择一个模块	单击要选择的模块，当用户选择了一个新的模块后，之前选择的模块被放弃
选择多个模块	按住鼠标左键不放拖动鼠标光标，将要选择的模块包括在鼠标光标画出的方框里；或者按住 Shift 键，然后逐个选择
不同窗口间复制模块	直接将模块从一个窗口拖动到另一个窗口
同一模型窗口内复制模块	先选中模块，然后按下 Ctrl+C 组合键，再按下 Ctrl+V 组合键；还可以在选中模块后，通过快捷菜单来实现
移动模块	按下鼠标左键直接拖动模块
删除模块	先选中模块，再按下 Delete 键
连接模块	先选中源模块，然后按住 Ctrl 键并单击目标模块
断开模块间的连接	先按下 Shift 键，然后用鼠标左键拖动模块到另一个位置；也可以将鼠标光标指向连线的箭头处，当出现一个小圆圈圈住箭头时按下鼠标左键并移动连线
改变模块大小	先选中模块，然后将鼠标光标移到模块方框的一角，当鼠标光标变成两端有箭头的线段时，按下鼠标左键拖动模块图标以改变图标大小
调整模块的方向	先选中模块，然后通过 Format→Rotate Block 命令来改变模块方向
给模块加阴影	先选中模块，然后通过 Format→Show Drop Shadow 命令来给模块加阴影
修改模块名	双击模块名，然后修改
模块名的显示与否	先选中模块，然后通过 Format→ShowName/Hide Name 命令来决定是否显示模块名
改变模块名的位置	先选中模块，然后通过 Format→Flip Name 命令来改变模块名的显示位置
在连线之间插入模块	用鼠标光标将模块拖动到连线上，使得模块的输入/输出端口对准连线

表 14-13　对直线进行操作

任　　务	Microsoft Windows 环境下的操作
选择多条直线	与选择多个模块的方法一样
选择一条直线	单击要选择的连线，当用户选择一条新的连线时，之前选择的连线被放弃
连线的分支	按下 Ctrl 键，然后拖动直线；或者按下鼠标左键并拖动直线
移动直线段	按下鼠标左键直接拖动直线
移动直线顶点	将鼠标光标指向连线的箭头处，当出现一个小圆圈圈住箭头时按下鼠标左键移动连线
直线调整为斜线段	按下 Shift 键，将鼠标光标指向需要移动的直线上的一点并按下鼠标左键直接拖动直线
直线调整为折线段	按住鼠标左键不放直接拖动直线

Simulink 中几乎所有模块的参数都允许用户进行设置，只要双击要设置的模块或在模块上单击鼠标右键，并在弹出的快捷菜单中选择相应模块的参数设置命令，就会弹出模块参数对话框。该对话框分为两部分，上面一部分是模块功能说明，下面一部分用来进行模块参数设置。

同样，双击要设置的模块，就会弹出参数设置对话框，如图 14-19 所示。每个模块的参数设置对话框都会有对自身功能描述的部分。

图 14-19 中显示了一个增益模块的配置。用户可以设置增益的增益大小、采样时间和输出数据类型等参数。模块参数还可以通过 set_param 命令设置。

Simulink 中的每个模块都有一个内容相同的属性（Properties）设置对话框，用鼠标右键单击模块并在弹出的快捷菜单中选择 Block Properties 命令，或双击要设置的模块，就会弹出属性设置对话框，如图 14-20 所示。

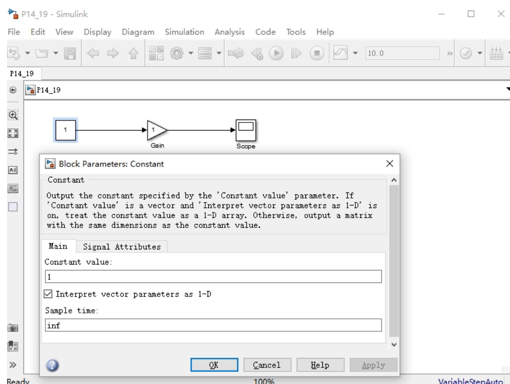


图 14-19　模块参数设置对话框

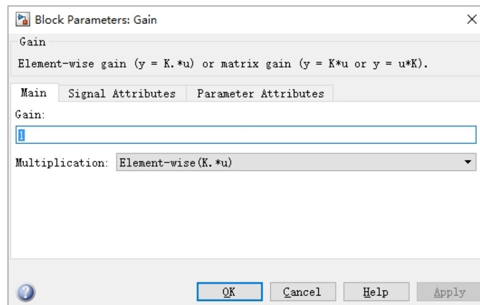


图 14-20　模块属性设置对话框

属性设置对话框主要包括 3 项内容。

1．General 选项卡

- Description：用于对该模块在模型中的用法进行注释。
- Priority：规定该模块在模型中相对于其他模块执行的优先顺序。优先级的数值必须是整数。如果用户不输入数值，系统就会自动选取合适的优先级。优先级的数值越小（可以是负整数），优先级越高。一般不需要设置它。

- Tag：用户为模块添加的文本格式的标记。

2．Block Annotation 选项卡

Block Annotation 选项卡用于指定在模块的图标下显示模块的哪个参数及其值，以及以什么格式显示。属性格式字符串由任意的文本字符串加嵌入式参数名组成。选中需要设置的模块，单击鼠标右键在弹出的快捷菜单中选择 properties，就可以打开如图 14-21 所示的窗口。

首先设置 General 选项卡的 Priority 为 4，然后在 Block Annotation 选项卡的 Enter text and tokens for annotation 文本框中输入 Priority=%<priority>\n Gain=%<gain>，最后单击 Apply 按钮，注意观察模块图标的变化。

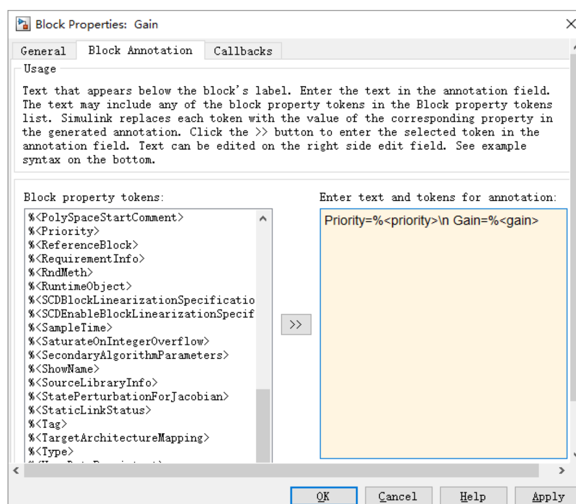


图 14-21 设置 Block Annotation

3．Callbacks 选项卡

Callbacks 选项卡用于定义当该模块发生某种特殊行为时所要执行的 MATLAB 表达式，也称回调函数。关于回调函数，在后面的章节再进行介绍。

对信号进行标注及在模型图表上建立描述模型功能的注释文字，是一个好的建模习惯。信号标签和注释如图 14-22 所示。表 14-14 和表 14-15 列出了对标注和注释的具体操作方法。

给一个信号添加标签，只需在直线上双击，然后输入文字即可。建立模型注释与之类似，只要在模型窗口的空白处双击，然后输入注释文字即可。

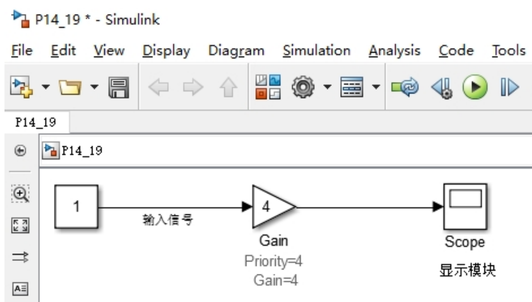


图 14-22 注释和信号标注示例

表 14-14　对标注进行处理

任　务	Microsoft Windows 环境下的操作
建立信号标签	在直线上直接双击，然后输入
复制信号标签	按下 Ctrl 键，然后按住鼠标左键选中标签并拖动
移动信号标签	按住鼠标左键选中标签并拖动
编辑信号标签	在标签框内双击，然后编辑
删除信号标签	按下 Shift 键，然后单击选中标签，再按 Delete 键
用粗线表示向量	选择 Foamat→Port/SignalDisplays→WideNonscalarLines 命令
显示数据类型	选择 Foamat→Port/SignalDisplays→PortDataTypes 命令

表 14-15　对注释进行处理

任　务	Microsoft Windows 环境下的操作
建立注释	在模型图标中双击，然后输入文字
复制注释	按下 Ctrl 键，然后选中注释文字并拖动
移动注释	选中注释并拖动
编辑注释	单击注释文字，然后编辑
删除注释	按下 Shift 键，选中注释文字，再按 Delete 键

14.2.2　模型和模型文件

1．Simulink 模型的概念

Simulink 意义上的模型根据表现形式的不同有着不同的含义：在模型窗口中表现为可见的方框图；在存储形式上则为扩展名为.mdl 的 ASCII 文件；从其物理意义上讲 Simulink 模型模拟了物理器件构成的实际系统的动态行为。采用 Simulink 软件对一个实际动态系统进行仿真，关键是建立起能够模拟并代表该系统的 Simulink 模型。

从系统组成上来看，一个典型的 Simulink 模型一般包括 3 个部分：输入、系统及输出。系统也就是指在 Simulink 当中建立并研究的系统方框图；输入一般用信源（Source）表示，具体形式可以为常数、正弦信号、方波及随机信号等，代表实际对系统的输入信号；输出则一般用信宿（Sink）表示，具体可以是示波器、图形记录仪等。无论是输入、输出还是系统，都可以从 Simulink 模块中直接获得，或由用户根据实际需要来用模块组合而成。

当然，对一个实际的 Simulink 模型而言，这 3 种结构并不是必需的。有些模型可能不存在输入或输出部分。

2．模型文件的创建和修改

模型文件是指在 Simulink 环境当中记录模型中的模块类型、模块位置及各个模块相关参数等信息的文件，其文件扩展名为.mdl。换句话说，在 Simulink 中创建的模型是由模型文件记录下来的。在 MATLAB 环境中，可以创建、编辑并保存创建的模型文件。

3．模型文件格式

前面的示例模型都是通过图形界面来建立的，Simulink 还为用户提供了通过命令行建立模型和设置模型参数的方法。一般情况下，用户不需要使用这种方式来建模，因为它要求用户熟悉大量的命令，而且很不直观。这一节只对 Simulink 的模型文件格式做粗略的介绍，用户若在实际建模时遇到相关的问题，可以查阅在线帮助。

Simulink 将每一个模型（包括库）都保存在一个以.mdl 为扩展名的文件里，称为模型文件。一个模型文件就是一个结构化的 ASCII 文件，它包括关键字和各种参数的值。下面以一个示例来介绍如何查看模型文件和模型文件的结构。

典型的模型文件如下所示：

```
Model {
<Model Parameter Name><Model Parameter Value>
...
BlockDefaults {
<Block Parameter Name><Block Parameter Value>
...
}
AnnotationDefaults {
<Annotation Parameter Name><Annotation Parameter Value>
...
}
System {
<System Parameter Name><System Parameter Value>
...
Block {
<Block Parameter Name><Block Parameter Value>
...
}
Line {
<Line Parameter Name><Line Parameter Value>
Branch {
<Branch Parameter Name><Branch Parameter Value>
...
}
}
Annotation {
<Annotation Parameter Name><Annotation Parameter Value>
...
}
}
```

文件分成下面几个部分来描述模型。

- Model 部分：用来描述模型参数，包括模型名称、模型版本和仿真参数等。
- BlockDefaults 部分：用来描述模块参数的默认设置。
- AnnotationDefaults 部分：用来描述模型的注释参数的默认值，这些参数值不能用 set_param 命令来修改。
- System 部分：用来描述模型中每一个系统(包括顶层的系统和各级子系统)的参数。每一个 System 部分都包括模块、连线和注释等。

14.2.3　创建模型的基本步骤和技巧

通过前面的学习，读者可能觉得使用 Simulink 建模实在是太简单了，只不过是用鼠标来选择几个模块，然后再把它们用几条线连接起来，最后按一下“运行”按键观察结果曲线就可以了。

但是当再次遇到实际问题时，读者可能会意识到实际工程中要考虑的方面非常复杂，因此读者需要进一步学习和掌握 Simulink 中更为深层次的内容。不过只要掌握了前面的内容，读者就可以通过在线帮助来解决更为复杂的问题了。

1．建模的基本步骤

在前面内容的基础上，总结出使用 Simulink 进行系统建模和系统仿真的一般步骤。

[1] 画出系统草图。将所要仿真的系统根据功能划分成一个个小的子系统，然后用一个个小的模块来搭建每个子系统。这一步骤也体现了用 Simulink 进行系统建模的层次性特点。当然所选用的模块最好是 Simulink 库里现有的模块，这样用户就不必进行烦琐的代码编写，当然这要求用户必须熟悉这些库的内容。

[2] 启动 Simulink 模块库浏览器，新建一个空白模型。

[3] 在库中找到所需模块并拖到空白模型窗口中，按系统草图的布局摆放好各模块并连接各模块。

[4] 如果系统较复杂、模块太多，可以将实现同一功能的模块封装成一个子系统，使系统的模型看起来更简洁（封装的方法在后面介绍）。

[5] 设置各模块的参数及与仿真有关的各种参数（在后面的相关小节有详细介绍）。

[6] 保存模型，模型文件的扩展名为.mdl。

[7] 运行仿真，观察结果。如果仿真出错，请按弹出的错误提示框来查看出错的原因，然后进行修改；如果仿真结果与预想的结果不符，首先要检查模块的连接是否有误、选择的模块是否合适，然后检查模块参数和仿真参数的设置是否合理。

[8] 调试模型。如果在上一步中没有检查出任何错误，那么就有必要进行调试，以查看系统在每一个仿真步骤的运行情况，直至找到出现仿真结果与预想的或实际情况不符的地方，修改后再进行仿真，直至结果符合要求。当然最后还要保存模型。

2．建模的方法与技巧

当创建的模型较为复杂时，可以通过将相关的模块组织成子系统来简化模型的显示。

使用子系统具有的优点如下：

- 有助于减少模型窗口中显示的模型数量。
- 允许用户将模型中功能相关的模块组织在一起。
- 使用户可以建立分层的模型框图。其中，子系统模块位于某个层次，而组成子系统的模块则位于另外的层次。

创建子系统的方法大致可分为两种：一种是在模型中加入一个子系统（Subsystem）模块，然后打开子系统窗口并进行编辑；另一种则是直接选中组成子系统的模块，然后单击相

应的菜单完成系统的创建。具体创建方法将在后面详细描述。

通过子系统，用户可以创建分层的模型结构，还可以使用模型浏览器来浏览模型中的层次关系。

Simulink 会在子系统模块上标明端口。标识的内容一般是子系统的输入或输出模块名称，子系统正是通过这些模块与子系统外的模块进行交互的。用户可以通过下面的操作隐藏某些或所有的端口标识。

- 选择子系统模块，单击 Format:Hit Port Lables 按钮来隐藏所有端口标识。
- 在子系统中选择某个输入或输出模块，然后单击 Format:Hit Port Lables 按钮来隐藏相应端口标识。
- 在子系统模块的对话框中检查 showPort Lables 栏。

Simulink 允许用户设置访问子系统的权限，禁止使用者观看或修改子系统的内容，这可以通过在子系统属性对话框的 Access 栏中设置相应的参数值来实现。用户可以将它设置成 ReadOnly 或 NoReadorWrite，其中 ReadOnly 参数表示允许查看局部拷贝子系统的内容，但禁止修改原始拷贝中的内容；NoReadorWrite 参数表示禁止对子系统的查看、修改或复制。

3．建模时的考虑

- 内存因素：一般而言，计算机的内存越大，Simulink 运行得越好。
- 使用分层：如果系统的模型比较复杂，使用子系统的分层机制将会大大简化模型，用户可以将主要精力放在系统信号的主要走向上，而忽略其中各个功能模块的实现细节。同时，也方便模型的理解和阅读。
- 使用注释：一般而言，具有详细的注释和帮助信息可以让模型更容易阅读和理解。因此，在模型创建过程中，应尽可能为信号添加注释。另外，为模型添加说明信息，可以让创建的模型更容易阅读。
- 建模策略：用户在创建模型时可以发现，如果模型小的模块重复比较多，那么模型的保存也相对容易。可以将反复用到的模块全部放在新建的模块库中，这样可以在 MATLAB 窗口中通过模块的名字来访问它们。

一般来说，在创建模型前，可以首先在纸上将系统的方框图设计好，然后将需要用到的模块改在模型窗口当中，最后完成模块的连线。这样可以避免频繁地打开模块库。

4．方程的建模

要知道，仿真的系统一般是采用数学方程来描述的，因此在模型的创建过程中，经常遇到的是对方程的建模，而初学者往往对此感到疑惑。下面通过实例来说明方程建模的一般步骤。

例 14-1，模拟一次线性方程：

$$y=9/4(x)+30$$

其中输入信号 x 是幅值为 10 的正弦波。

（1）建模所需模块的确定。

在进行建模之前，首先要确定建立上述模型需要的模块。



- 一个 Gain 模块：用于定义常数增。Gain 模块来源于 Math Library。
- 一个 Constant 模块：用于定义一个常数。Constant 模块来源于 Source Library。
- 一个 Sum 模块：用于将两项相加。Sum 模块来源于 Math Library。
- —个 Sine Wave 模块：用于作为输入信号。Sine Wave 模块来源于 Source Library。
- 一个 Scope 模块：用于显示系统输出。Scope 模块来源于 Sinks Library。

（2）模块的复制。

把上面这些模块从各自的模块库中复制到用户的模型窗口，如图 14-23 所示。

分别打开 Gain 模块和 Constant 模块（双击模块图标），分别将它们设置为 9/4 和 30，然后单击 OK 按钮。打开 Sine Wave 模块，把它的幅值设为 10，以使其得到较大的变化。

（3）模块的连接。

把各个模块连接起来，得到如图 14-24 所示的方框图。

Sine Wave 模块代表摄氏温度，Gain 模块输出为 9/4，这个值在 Sum 模块和 Constant 模块中的常数 30 相加后得到输出，这个输出就是 *y*。打开 Scope 模块就可以观看这个输出值的变化曲线，其中，Scope 模块的 *x* 轴设为比较小的时间，如 10s，而把 *y* 轴设置得比幅值略大一些，以便能够得到整条曲线。

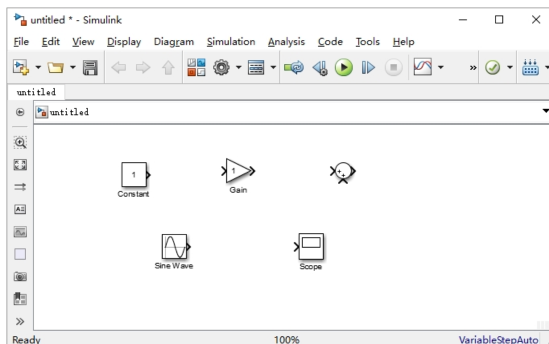


图 14-23　模块图

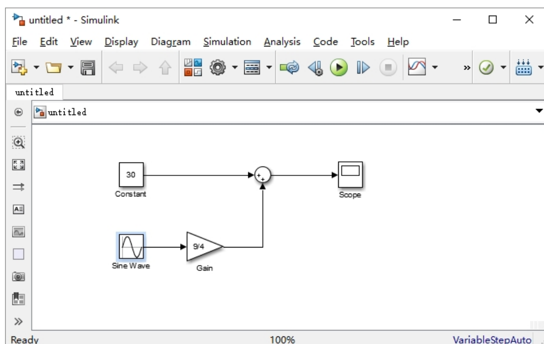


图 14-24　连线图

（4）开始仿真。

在模型窗口中选择 Simulation→Configuration Parameters 命令，定义 Stop time 为 10s，Max step size 为 0.1s，然后选择 Simulation→Start 命令，仿真开始，仿真曲线如图 14-25 所示。

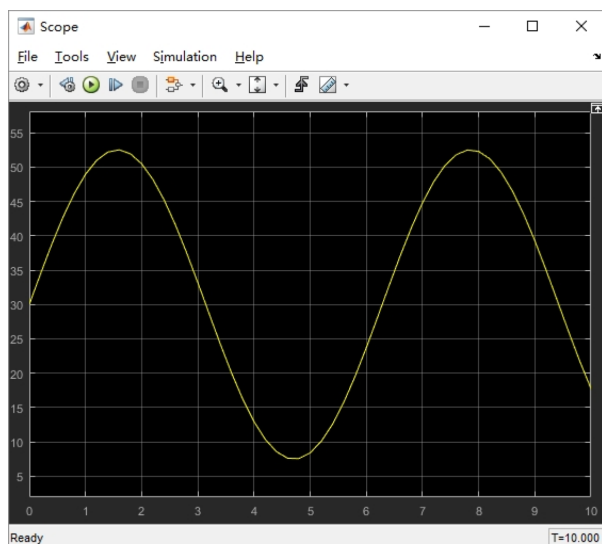


图 14-25　仿真曲线

14.3　过零检测和代数环

当仿真一个动态系统时，Simulink 在每一个时间步使用过零检测技术来检测系统状态变量的间断点。如果 Simulink 在当前的时间步内检测到了不连续的点，那么它将找到发生不连续的精确时间点，并且会在该时间点的前后增加附加的时间步。

有些 Simulink 模块的输入端口是支持直接输入的，这意味着这些模块的输出信号值在不知道输入端口的信号值之前不能被计算出来。当一个支持直接输入的输入端口由同一模块的输出直接或间接地通过由其他模块组成的反馈回路的输出驱动时，就会产生一个代数环。

本节将向读者介绍过零检测的工作原理及如何产生代数环。

14.3.1　过零检测

1．过零检测的工作原理

使用过零检测技术，一个模块能够通过 Simulink 注册一系列过零变量，每一个变量就是一个状态变量（含有不连续点）的函数。当相应的不连续发生时，过零函数从正值或负值传递零值。在每一个仿真步结束时，Simulink 通过调用每一个注册了过零变量的模块来更新变量。然后 Simulink 检测是否有变量的符号发生改变（相对于上一仿真时间点的结果），如果有改变就说明当前时间步有不连续发生。

如果检测到过零点，Simulink 就会在每一个发生符号改变的变量的前一时刻值和当前值之间插入新值以评估过零点的个数，然后逐步增加内插点数目并使其值依次越过每一个过零点。通过该技术，Simulink 可以避免精确地在不连续发生点进行仿真（不连续点处的状态变量的值可能没有定义）。

过零检测使得 Simulink 可以精确地仿真不连续点而不必通过减小仿真步长、增加仿真点

来实现，因此仿真速度不会受太大影响。大多数 Simulink 模块都支持过零检测，表 14-16 列出了 Simulink 中支持过零检测的模块。如果用户需要显示定义的过零事件，可以使用 Discontinuities 子库中的 Hit Crossing 模块来实现。

如果仿真的误差容忍度设置得太大，那么 Simulink 有可能检测不到过零点。例如，如果在一个时间步内存在过零点，但是在时间步的开始和最终时刻没有检测到符号的改变（如图 14-26 所示），那么求解器将检测不到过零点。

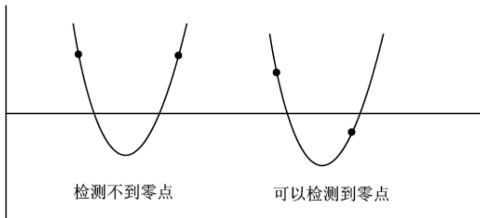


图 14-26　过零点检测

表 14-16　支持过零点检测的模块

模块名	说　明
Abs	一个过零检测：检测输入信号沿上升或下降方向通过零点
Backlash	两个过零检测：一个检测是否超过上限阈值，一个检测是否超过下限阈值
Dead Zone	两个过零检测：一个检测何时进入死区，一个检测何时离开死区
Hit Crossing	一个过零检测：检测输入何时通过阈值
Integrator	若提供了 Reset 端口，就检测何时发生 Reset；若输出有限，则有 3 个过零检测，即检测何时达到上限饱和值、检测何时达到下限饱和值和检测何时离开饱和区
MinMax	一个过零检测：对于输出向量的每一个元素，检测一个输入何时成为最大值或最小值
Relay	一个过零检测：若 relay 是 off 状态，就检测开启点；若是 on 状态，就检测关闭点
Relational Operator	一个过零检测：检测输出何时发生改变
Saturation	两个过零检测：一个检测何时达到或离开上限，一个检测何时达到或离开下限
Sign	一个过零检测：检测输入何时通过零点
Step	一个过零检测：检测阶跃发生时间
Switch	一个过零检测：检测开关条件何时满足
Subsystem	用于有条件地运行子系统：一个使能端口，一个触发端口

2．过零检测的重要性

在动态系统的运行过程中，状态的不连续经常会发生重要的事件，如当一个小球与地面发生碰撞时，它的位置就会产生急剧的变化。不连续常常会导致动态系统的显著变化，因此对不连续点进行精确的仿真非常重要，否则会导致仿真得到错误的系统行为。例如，仿真碰撞中的小球，如果小球与地面碰撞的时间点发生在仿真时间步内，模型中的小球会在半空中改变方向，这会使研究者得到与物理学上的规律相违背的结论。

为了避免得到错误的结论，使不连续点发生的时刻成为仿真的一个时间点很重要。对于一个纯粹靠求解器来决定仿真时间的仿真器很难有效地做到上面的要求。

固定步长的求解器在整数倍时间步长的时间点上计算状态变量的值，然而这并不能保证仿真时间点发生在状态不连续的时间点。当然，用户可以通过减小时间步长使状态不连续的时间点成为一个仿真时间点，但是这会减慢仿真速度。

可变步长的求解器可以保证有仿真时间点发生在状态不连续的时间点。它可以动态地改变步长，当状态变量变化慢时增加步长，变化快时减小步长。在不连续点附近，系统状态发生剧烈变化，因此理论上这种求解器可以精确地找到不连续发生的时间点，但是这会增加仿真的时间点，从而使仿真速度变慢。

14.3.2　代数环

从代数的角度来看，如图 14-27 所示的模块的解是 $z = 1$，但是大多数的代数环是无法直接看出解的。Algebraic Constraint 模块为代数方程等式建模及定义其初始猜想值提供了方便，它约束输入信号 $F(z)$等于零并输出代数状态 z，其输出必须能够通过反馈回路影响输入。用户可以为代数环状态提供一个初始猜想值，以提高求解代数环的效率。

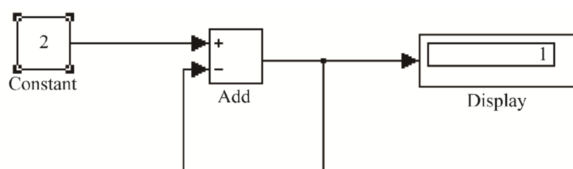


图 14-27　代数环

一个标量代数环代表一个标量等式或者一个形如 $F(z) = 0$ 的约束条件，其中 z 是环中一个模块的输出，函数 F 由环路中的另一个反馈回路组成。可将如图 14-27 所示的含有反馈环的模型改写成用 Algebraic Constraint 模块创建的模型，其仿真结果不变，如图 14-28 所示。

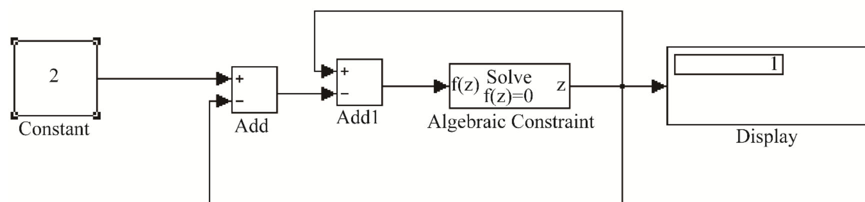


图 14-28　Algebraic Constraint 模块创建代数环模块

创建向量代数环也很容易，在如图 14-29 所示的向量代数环中可用下面的代数方程描述：

$$z2 + z1 - 2 = 0$$
$$z2 - z1 - 2 = 0$$

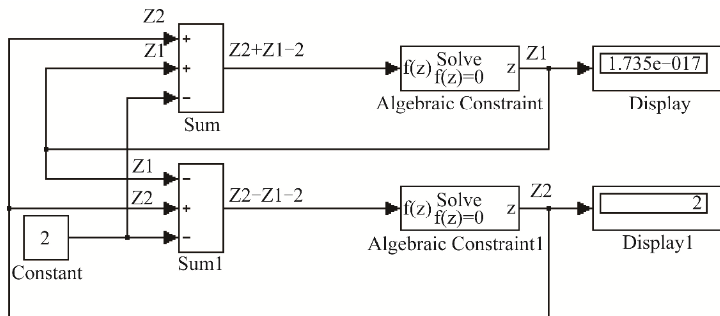


图 14-29　向量代数环

当一个模型包含一个 Algebraic Constraint 模块时就会产生一个代数环，这种约束可能是系统的物理连接的结果，也可能是用户试图为一个微分-代数系统（DAE）建模的结果。

为了求解 $F(z) = 0$，Simulink 环路求解器会采用弱线性搜索的秩为 1 的牛顿方法更新偏微分 Jacobian 矩阵。尽管这种方法很有效，但是如果代数状态 z 没有一个好的初始估计值，解法器有可能不收敛。

此时用户可以为代数环中的某个连线（对应一个信号）定义一个初始值，设置的办法有两种：可以通过 Algebraic Constraint 模块的参数设置；还可以通过在连线上放置 IC 模块（初始信号设置模块）来设置。

当一个系统包含代数环时，Simulink 会在每一个时间步进行循环求解。如果有可能，环路求解器会采用迭代的办法来求解，因此仿真速度会比较慢。

当一个模型中含有 Atomic Subsystem、Enabled Subsystem 或 Model 模块时，Simulink 可以通过模块的参数设置来消除其中一些代数环。对于含有 Atomic Subsystem Enabled Subsystem 模块的模型，可在模块参数设置对话框中选择 Minimize algebraic loop occurrences 项；对于含有 Model 模块的模型，可在 Configuration Parameters 对话框的 Model Referencing 面板中选择 inimize algebraic loop occurrences 项。

14.4　本章小结

本章主要介绍了使用 Simulink 进行仿真的基础知识，包括 Simulink 仿真相关的基本概念、工作环境、模型创建、过零检测及代数环等内容。这些内容是进行 Simulink 仿真的基础，在第 15 章中将继续介绍 Simulink 仿真的基础应用。

第 15 章

Simulink 仿真应用

知识要点

前面介绍了 Simulink 仿真的基础知识，本章将介绍使用 Simulink 进行仿真的基础应用。本章将涉及子系统、模型分析、运行仿真、模型调试等方面的知识和应用，与第 14 章结合起来，构成 Simulink 仿真的基础知识和应用体系。本书只对 Simulink 做简单的讲解，更深的内容请参考 MATLAB 帮助文件。

学习要求	知识点 \ 学习目标	了解	理解	应用	实践
	Simulink 仿真模型的分析	√			√
	Simulink 系统模型及其特点			√	√
	Simulink 系统仿真与调试		√		√
	Simulink 模型的基本调试方法		√		√



15.1　子系统的创建和封装

Simulink 在创建系统模型的过程中常常采用分层的设计思想。用户在进行动态系统的建模过程中，可以根据需要将模型中比较复杂，或者共同完成某一功能的基本模块（也可能是低一层次的子系统）封装起来，并采用一个简单的图形来替代表示。这样可使整个模型结构清晰、显示简单，让用户将主要精力放在系统的信号分析上。这种设计方法实际上也是面向对象设计思想的一种体现。

依照封装后系统的不同特点，Simulink 具有一般子系统、封装子系统和条件子系统 3 种不同类型的子系统，下面将分别对各种子系统的基本特点和创建过程依次给予介绍。

15.1.1　子系统介绍

1．分层的建模思想

用户在进行动态系统的仿真过程中，常常会遇到比较复杂的系统。无论多么复杂的系统都是由众多不同的基本模块组成的。在采用 Simulink 创建系统模块时，当然希望将系统所有的模块都在一个模型窗口中显示出来，然而对于比较庞大的系统，这是不现实的。

另外，较大的系统一般包含大量的基本模块，用户一般并不关心这些基本模块之间的信号交互，只希望了解系统不同组成部分之间的信号流向。因此，有必要根据系统的结构，将同属于一个部分的基本模块封装起来，在模型窗口中用封装后的模块（极为简单）来替代原来的部分。这种思想被称为“分层建模”的设计方法，在实际工作中经常用到。

Simulink 本身就体现了“分层建模”的思想。例如各种基本模块库可看成是封装了相关基本模块的子系统。采用这种设计思想的优点如下：

- 体现了面向对象的设计方法。封装后的子系统对用户而言是不透明的，用户可将它看作一个“黑匣子”，只需要关心子系统两端的输入/输出，而不需要知道子系统内部信号的处理过程。
- 提高了效率和可靠性，封装后的子系统可实现“重用性”。
- 对于封装的子系统，其工作区与基本工作区相互独立，简化了模型的设计。
- 符合实际系统分层组成的实际情况。

依照封装系统的不同特点，Simulink 具有一般子系统、封装子系统、使能子系统和触发子系统 4 种类型的子系统。

2．用户模块库的定制

通过子系统的封装，用户可以很容易实现 Simulink 模块库的定制。将用户自己创建的模块封装起来，并给予封装后的模块库适当的外观和参数，就完成了用户自己的模块库的定制。在使用上，定制的模块库与标准的 Simulink 模块库并无区别。这样，用户就用自己的模块库对 Simulink 进行了扩展，Simulink 这种广泛的开放性无疑得到了越来越多用户的支持和喜爱。

3．条件子系统

在 Simulink 模块库中有两个特殊的模块：使能模块（Enable）和触发模块（Trigger）。

如果把这两个模块放在某个子系统中，则该子系统是否发生作用就取决于外界的条件是否满足，这种系统称为条件执行子系统。这种子系统为创建更加复杂系统的仿真模型提供了方便。

条件子系统又可分为位能子系统（Enable Subsysterm）、触发子系统（Trigger Subsysterm）及触发使能子系统（Enable and Trigger Subsysterm）。

4．一般子系统

根据前面的介绍知道，在 Simulink 中，如果被研究的系统比较复杂，那么直接用基本模块构成的模型就比较庞大，模型中的主要流向不容易辨认。此时，若把整个模型按照实现功能或对应物理器件划分成几块，将有利于对整个系统的概念抽象。创建一般子系统是其中最为简单的方法。

一般子系统与封装子系统不同，它只是在视觉上将整个模型进行分层，子系统内部的模块仍然共享 MATLAB 的基本工作区。由于不需要进行封装对话框的设置，一般子系统的创建过程比较简单。用户可以根据需要采用标准的 Simulink 模块进行，也可以在模型上直接框选。

5．采用框选法创建一般子系统

如果用户已经建立起系统的模型方框图，就可以采用直接框选法制作一般子系统（Encapsulated Subsystem）。具体步骤如下：

[1] 生成子系统：在模型窗口中，将需要包含进子系统的模块框起来（在图中将以虚线框标识），选择 Edit→Create Subsystem 命令，便将框选的部分包装在一个名为 Subsystem 的模块中。包装后，新生成子系统中的模块和信号线可能会显得比较杂乱，可以通过鼠标操作进行整理。

[2] 模块名的修改：双击新生成子系统中的模块名，将模块名改为其他适当的名称。

[3] 输入/输出端口的设置：双击新生成子系统的图标，弹出该子系统的结构模型窗口。把该结构模型窗口中的输入口模块的默认名改为其他适当的名称，把输出口模块的默认名也改为其他适当的名称。

[4] 选择 File→Save 命令，将以上创建的一般子系统保存起来。以后用户打开该模型文件时，Simulink 将只显示一般子系统。用户双击子系统图标，可以进一步观察其内部的构造。

15.1.2　创建子系统

在 Simulink 中有两种创建子系统的方法。

- 通过子系统模块来创建子系统：先向模型中添加 Subsystem 模块，然后打开该模块并向其中添加模块。
- 组合已存在的模块创建子系统。

下面通过两个示例来介绍这两种建立子系统的方法。

例 15-1，通过 Subsystem 模块来创建子系统。具体步骤如下：

[1] 从 Ports&Subsystems 中复制 Subsystem 模块到自己的模型中，如图 15-1 所示。

[2] 双击 Subsystem 模块图标，打开 Subsystem 模块编辑窗口。

[3] 在新的空白窗口创建子系统，然后保存。

[4] 运行仿真并保存。



图 15-1　通过 Subsystem 模块创建子系统

例 15-2，组合已存在的模块创建子系统。具体步骤如下：

[1] 创建如图 15-2 所示的系统。

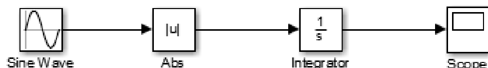


图 15-2　组合已存在的模块创建子系统

[2] 选中要创建成子系统的模块。

[3] 选择 Edit→Create Subsystem 菜单命令，结果如图 15-3 所示。

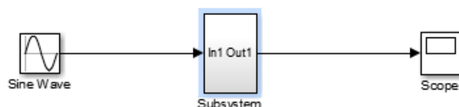


图 15-3　创建子系统示例

[4] 运行仿真并保存。

15.1.3　封装子系统

一般子系统的操作过程很简单，并且在一定程度上提高了分析问题的抽象能力。然而，一般子系统仍然从 MATLAB 基本工作中获取变量，因此没有实现完全意义上的封装。

在一般子系统的基础上对子系统进行封装，则可以解决这个问题。封装子系统在外表上与普通模块完全一样，有自己的参数对话框和图标，并且其中的变量有着与基本工作区独立的工作区，即封装工作区。

封装子系统是在一般子系统的基础上进一步设置而成的，封装子系统还为用户提供了一种扩展 Simulink 模块库的方法。

1．为何封装

封装子系统可以为用户带来如下好处：

- 在设置子系统中各个模块的参数时只通过一个参数对话框就可以完成所需设置。
- 为子系统创建一个可以反映子系统功能的图标。
- 可以避免用户在无意中修改子系统中模块的参数。

2．如何封装

用户可以按照如下步骤来封装一个子系统。

- 选择需要封装的子系统。
- 单击鼠标右键，在弹出的快捷菜单中选择 Mask→Create Mask 命令，这时会弹出如图 15-4 所示的封装编辑器，通过它进行各种设置。

单击 Apply 或 OK 按钮保存设置。

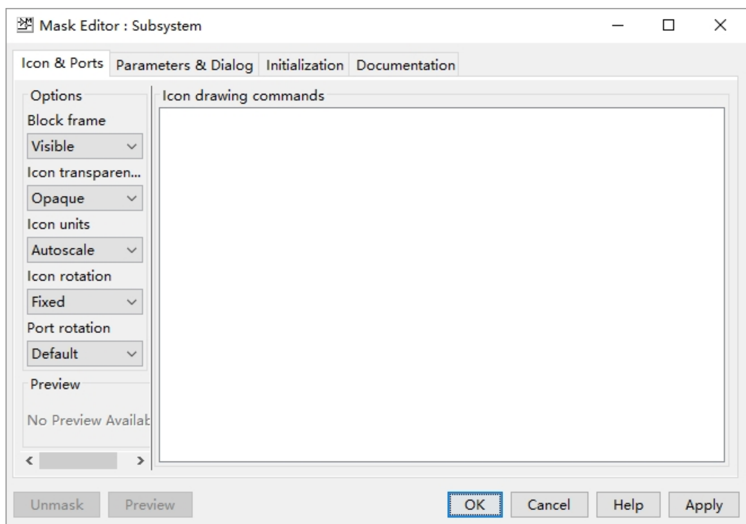


图 15-4　封装编辑器

3．封装示例

封装的操作很简单，也比较容易理解，下面通过一个简单的示例来介绍如何进行封装。

[1] 建立如图 15-5 所示的含有一个子系统的模型，并设置子系统中 Gain 模块的 Gain 参数为一个变量 m。

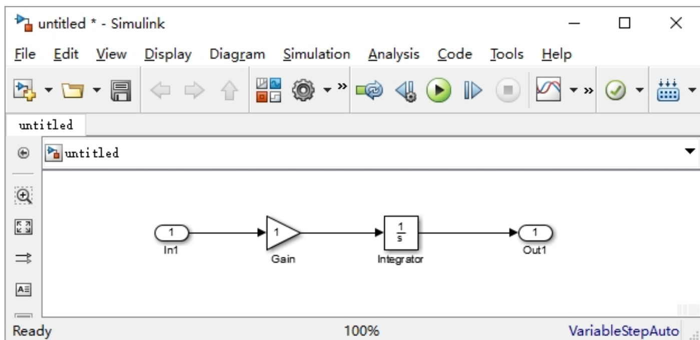


图 15-5　封装子系统示例

[2] 选中模型中的 Subsystem 子系统，用鼠标右键单击子系统，在弹出的快捷菜单中选择 Mask→Add Icon Image 命令，编辑子系统显示图形，如图 15-6 所示。



图 15-6　封装编辑器

[3] 进行封装设置。打开如图 15-4 所示的封装编辑器中的 Icon&Ports 选项卡，出现以下几个选项：

- Options 选项组：定义图标的边框是否可见（Frame）、系统在图标中自动生成的端口标签是否可见（Transparency）等，用户只要试一试就会很快理解和掌握。
- Icon drawing commands 文本框：用 MATLAB 命令来定义如何绘制模型的图标，这里的绘图命令可以调用 Initialization 选项卡中定义的变量。
- Preview 选项组：显示封装系统的图标预览。

[4] 设置 Parameters&Dialog 选项卡，如图 15-7 所示。

[5] 设置 Initialization 选项卡，如图 15-8 所示。

Initialization 选项卡允许用户定义封装子系统的初始化命令。初始化命令可以使用任何有效的 MATLAB 表达式、函数、运算符和在 Parameters 选项卡中定义的变量，但是初始化命令不能访问 MATLAB 工作区的变量。在每一条命令后用分号结束可以避免模型运行时在 MATLAB 命令行窗口显示运行结果。一般在此定义附加变量、初始化变量或绘制图标等。

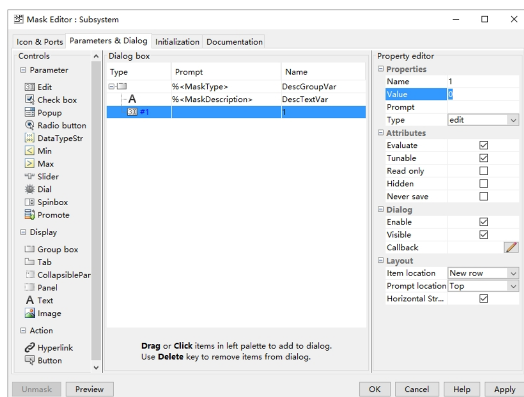


图 15-7　设置参数

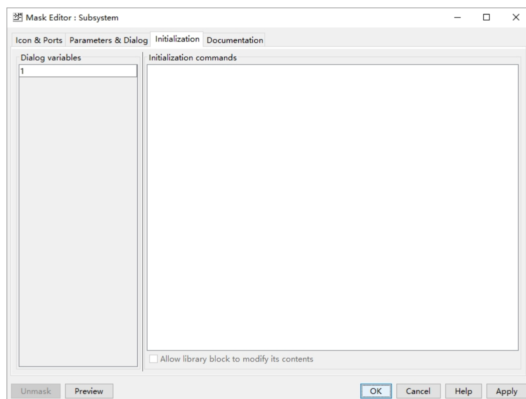


图 15-8　设置初始化参数

[6] 运行仿真，双击模型中的 Scope 模块，得到如图 15-9 所示的结果。

注意

如果在保存上面示例的模型时弹出 Error 提示框，则有可能是用户的 MATLAB 或 Simulink 在封装技术上不支持中文，这时需要将 Documentation 选项卡中的中文说明改成英文。

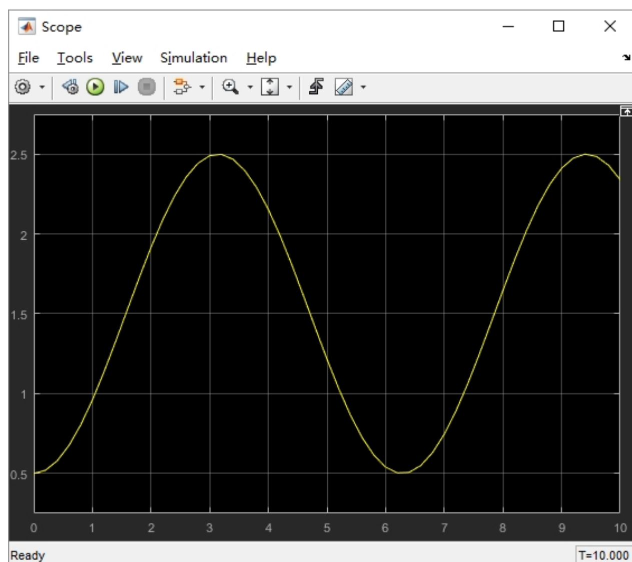


图 15-9　示例模型仿真结果

4．如何创建自己的模块库

当用户创建了很多封装子系统模块时，有必要分门别类地来存储这些模块。另外，在进行仿真建模时，为了减少到各子库中来回查寻所需模块，用户有必要将同一类功能的一组常用模块统一放置在同一模块库中。

通过选择 Simulink 用户界面的 File→New→Library 菜单命令来创建模块库。选中该命令后弹出一个空白的库窗口，然后将需要存放在同一模块库中的模块复制到模块库窗口中即可，如图 15-10 所示。创建好模块库后，用户在创建模型时可以不需要打开 Simulink 模块库浏览器，而只需要在 MATLAB 命令行窗口中输入存放相应模块的模块库的文件名即可。

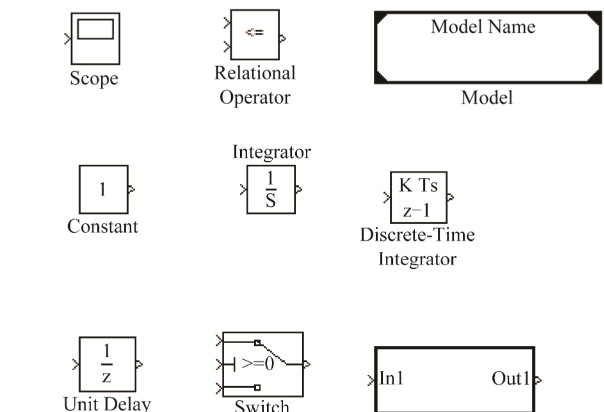


图 15-10　自建模块库

15.2　仿真模型的分析

用户在创建 Simulink 仿真模型之后，在启动仿真过程之前一般需要对所创建的模型进行分析，其目的是为了以后正确设置仿真的参数和配置。

所创建的模型究竟存在多少种状态？是否存在代数环？采用哪种微分方程求解器？积分步长和容许误差究竟取多大合适？这些问题对于提高系统模型的仿真质量（速度和精度）具有相当重要的意义，其中有些问题是初学者常常容易忽视而又经常遇到的，因此有必要对这些问题单独组织在一起进行专门论述。

15.2.1　模型状态的确定

在进行 Simulink 仿真过程中常常需要为仿真模型设置初始状态，然而对于实际比较复杂的系统模型，一般很难直接看出模型状态的个数及相关变量。尤其当用户在命令行窗口中运行仿真过程时，就必须事先知道系统模型中究竟有多少种状态，其中多少是连续量，多少是离散量，模型中的模块对应着哪一个状态分量。

从本质上而言，各种模块就是图形化的微分或差分方程。无论是高阶微分或差分方程，还是传递函数，Simulink 在其内部总是采用连续或离散状态方程加以描述。

对于简单的系统，可以直接根据模块的特点分析出模型的状态：积分模块直接对应着一个连续状态变量；单位延迟模块对应着一个离散状态；而传递函数和状态中间模块所对应的状态变量数目由模块本身的阶次决定。

对于不能直接看出状态的系统，可以在命令行窗口中输入相应的指令进行分析。

从模型获得状态信息的指令就是模型名称本身。具体格式如下：

```
[sys,x0,str,ts] = model([],[],[],'sizes');
```

其中，model 为具体模型名，size 是输出参数，它是一个 7 元向量，其中各元素的含义如下：

- 元素 1：状态向量中连续分量的数目。
- 元素 2：状态向量中离散分量的数目。
- 元素 3：输出分量的总数。
- 元素 4：输入分量的总数。
- 元素 5：系统中不连续解的数目。
- 元素 6：系统中是否含有直通回路。
- 元素 7：不同采样速率的类别数。

从上面可以看出，数组 size 包含该模块的许多基本特征。

*x*0 返回的是模型状态向量的初始值，可以通过对 *x*0 赋值来设置状态向量的初始值。

sys 按次序给出了所有状态变量对应模块的所在模型名称、子系统名及模块名。

下面通过例子来说明模型特征的获取方法。

例 15-3，以 MATLAB 的演示模型 ssc_house_heating_system.mdl 为对象确定模型的状态，如图 15-11 所示。

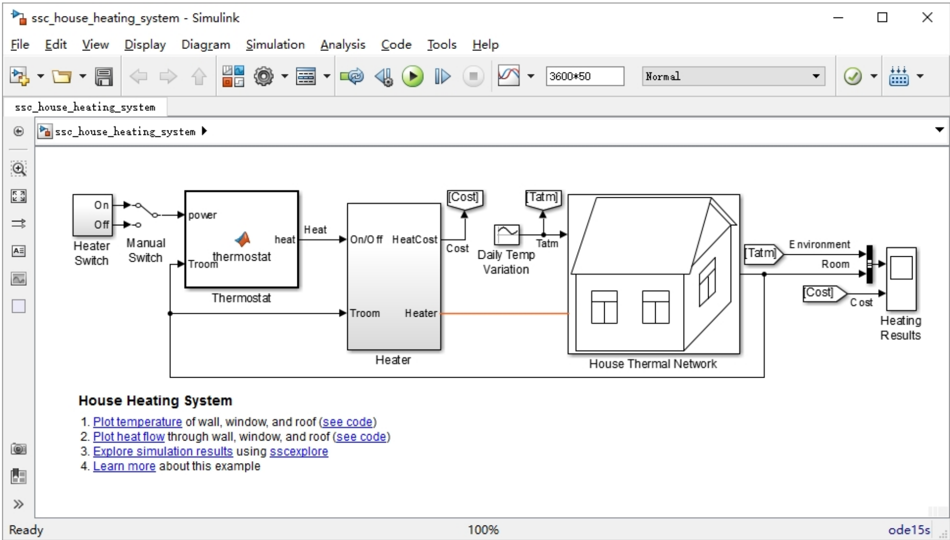


图 15-11　vdp 方程

代码如下：

```
>> [sys,x0,str,ts] = ssc_house_heating_system ([],[],[],'sizes')

sys =
    6
    4
    0
    0
    0
    0
    2
x0 =
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
str =
    'ssc_house_heating_system/House             Thermal             Network/Solver
Configuration/EVAL_KEY/STAT…'
    'ssc_house_heating_system/House             Thermal             Network/Solver
Configuration/EVAL_KEY/STAT…'
    'ssc_house_heating_system/House             Thermal             Network/Solver
Configuration/EVAL_KEY/STAT…'
    'ssc_house_heating_system/House             Thermal             Network/Solver
Configuration/EVAL_KEY/STAT…'
    'ssc_house_heating_system/Heater/Integrator'
```

```
        'ssc_house_heating_system/Heater/Transfer Fcn'
        'ssc_house_heating_system/House            Thermal              Network/Solver
Configuration/EVAL_KEY/INPU…'
        'ssc_house_heating_system/House            Thermal              Network/Solver
Configuration/EVAL_KEY/INPU…'
        'ssc_house_heating_system/House            Thermal              Network/Solver
Configuration/EVAL_KEY/INPU…'
        'ssc_house_heating_system/House            Thermal              Network/Solver
Configuration/EVAL_KEY/INPU…'

    ts =
        0    0
        0    1
```

结果显示，该模型中的积分模块 x1、x2 形成了两个连续状态。

15.2.2　线性化的数学描述

在非线性仿真的过程中，常常需要将非线性系统在工作点附近进行近似线性化，然后再利用线性化方法来研究非线性系统在工作点附近的一些动态特征。本节将介绍如何利用 Simulink 获取非线性系统的近似线性模型。

1. 模型的线性化问题

非线性系统可以通过如下的状态方程进行描述：

$$\begin{cases} \dot{x} = f(x,\boldsymbol{u},t) \\ \boldsymbol{y} = g(x,\boldsymbol{u},t) \end{cases}$$

式中，$\dot{x}$、$\boldsymbol{f}$ 都是 n 阶向量，$\boldsymbol{u}$ 是 m 阶向量，$\boldsymbol{y}$ 是 p 阶向量。在系统工作点附近进行近似线性化，只保留系统的一次项，而将系统的高次项都舍去，可以得到如下方程：

$$\begin{cases} \dot{x} = \boldsymbol{A}x + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} = \boldsymbol{C}x + \boldsymbol{D}\boldsymbol{u} \end{cases}$$

其中，各参数矩阵的计算如下：

$$\boldsymbol{A} = \frac{\partial}{\partial x} f(x,\boldsymbol{u},t)$$

$$\boldsymbol{B} = \frac{\partial}{\partial \boldsymbol{u}} f(x,\boldsymbol{u},t)$$

$$\boldsymbol{C} = \frac{\partial}{\partial x} g(x,\boldsymbol{u},t)$$

$$\boldsymbol{D} = \frac{\partial}{\partial \boldsymbol{u}} g(x,\boldsymbol{u},t)$$

这里的近似只是相对比较简单的系统在平衡点附近而言的，对于非线性特性较为复杂的系统，这样的近似可能会引起系统特性的较大变化。

2．为何线性化

一旦数据具有状态空间的形式或是被转换成 LTI 对象，用户就可以使用控制系统工具箱里的函数做进一步的分析。

- 将状态空间转换为 LTI 对象：sys = ss(A,B,C,D)。
- 绘制系统的波特相位幅频图：bode(A,B,C,D) 或 bode(sys)。
- 求系统的时间响应：step(A,B,C,D) 或 step(sys)（单位阶跃响应）；impulse(A,B,C,D) 或 impulse(sys)（单位脉冲响应）；lsim(A,B,C,D,u,t) 或 lsim(sys,u,t)（任意输入响应）。

3．如何线性化

用户可以通过下面几个函数对所建模型进行线性化。

- [A,B,C,D] = linmod('sys')。
- [A,B,C,D] = linmod('sys',x,u)。
- [A,B,C,D] = linmod('sys', x, u, para)。
- [A,B,C,D] = linmod('sys', x, u, 'v5', para)。
- [A,B,C,D] = linmod('sys', x, u, 'v5', para, xpert, upert)。
- [A,B,C,D] = dlinmod('sys', x, u)。
- [A,B,C,D] = dlinmod('sys',Ts, x, u, 'v5', para)。
- [A,B,C,D] = dlinmod('sys', x, u, 'v5', para, xpert, upert)。
- argout = linmod2('sys', x, u, para)。

其中，*x*、*u* 用来指定工作点处的系统状态和输入信号，默认均为适当的零向量；linmod 用于线性化本身是线性和非线性的模型；dlinmod 用于线性化本身是离散系统或者离散和连续的混合系统。对于以上函数的具体使用方法及其各个参数的含义，用户可以查询在线帮助。

15.2.3　平衡点分析

Simulink 通过 trim 命令来决定动态系统的稳定状态点。所谓稳定状态点就是满足用户自定义的输入/输出和状态条件的点。下面通过一个简单的示例来介绍如何使用 trim 命令求解平衡点。

例 15-4，建立如图 15-12 所示的模型，保存为 exb.mdl。

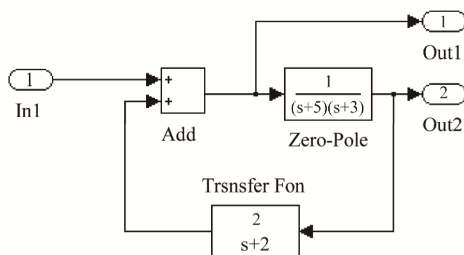


图 15-12　求解系统平衡点

为系统状态和输入定义一个初步的猜测值，并把用户预期的输出值赋给 y，代码设置如下：

```
>>x = [0;0;0];
>>u = 0;
>>y = [3;3];
```

使用索引变量规定模型的输入/输出和状态中哪些可变，哪些不可变，代码设置如下：

```
>>ix = [];
>>iu = [];
>>iy = [1;2];
```

使用 trim 命令来求出平衡点，代码设置如下：

```
>>[x,u,y,dx] = trim('exb',x,u,y,ix,iu,iy)
```

得到结果：

```
x =
   0.1875
   0.0000
   1.4524
u =
   5.2500
y =
   5.6250
   0.3750
dx =
  1.0e-015 *
       0
   0.8882
   0.0000
```

15.2.4　微分方程的求解算法

1．微分方程求解

目前数学上存在多种求解微分方程的算法，有的算法是很经典的。但在众多的求解算法中，一般不存在通用的、最好的求解方法，具体运用哪种方法需要具体情况具体分析。这就需要在选择求解方法之前仔细分析系统的动态性能。

Simulink 提供了多种求解微分方程的方法，如龙格-库塔法、阿达姆斯法、Gear 法、Euler 法、1insim 法等。Simulink 中还增加了其他包括求解两点边值问题的算法。

简单来讲，龙格-库塔法适合于高度非线性或不连续的系统，不适合刚性系统（既有快变特性又有慢变特性的系统）；阿达姆斯法适合于非线性不大、时间参数变化小的系统；Gear 法专门用于刚性系统的仿真，对非线性系统的仿真能力较差；Euler 法比较差，一般不采用；linsim 法适合于近似线性的系统，尤其适合于线性刚性系统。

2．各种求解方法的比较

下面针对 Simulink 中经常使用的几种求解算法进行分析。

（1）ODE45 和 ODE23。

ODE45 和 ODE23 两种方法都属于龙格-库塔法，都是用有限的 Taylor 级数去近似解函

数。有限项 Taylor 级数近似的主要误差是所谓的阶段误差（由截去的高阶项引起）。

ODE45 分别采用四阶和五阶 Taylor 级数计算每个积分步长最后的状态变量近似值，并把这两个近似值的差作为对阶段误差大小的估计。如果误差估计值比较大，那么就把该积分步长缩短，然后再重新进行计算，直到误差估计值小于指定的精度范围。如果误差估计值远小于指定精度，那么将把下一个积分步长缩短。

ODE23 在每一个积分步长中是采用二阶和三阶 Taylor 级数计算每个积分步长最后的状态变量近似值的。上面的论述表明 ODE23 和 ODE45 都是可变步长的。

（2）ODE113。

ODE113 是变步长的阿达姆斯法，属于多步预估校正法。

在预报阶段，ODE113 采用多项式近似，多项式的系数可以通过前面 n-1 个解及其导数值确定；然后采用外推法计算下一个解。在校正阶段，对前面 n 个解和新得到的解，通过拟合或校正多项式，最后用校正多项式更新计算解，从而对原有解进行校正。预测解和校正解之间的误差可以用来调整积分步长。

15.3　运行仿真

用户在创建系统的仿真框图并对它进行分析之后，就可以启动仿真过程了。Simulink 支持两种不同启动仿真的方法：直接从模型窗口中启动和在命令行窗口中启动。无论哪种方法，最终的仿真过程是相同的。在仿真启动之前，还需要仔细配置仿真的基本设置。如果有的设置不合理，仿真过程甚至不可能进行下去。

本节将讲述运行仿真之前的一些准备工作，包括仿真具体配置的实现和运行仿真的一般步骤、结果的显示及提高仿真速度和精度所采取的措施等，旨在使读者了解 Simulink 仿真的基本过程，学会 Simulink 仿真的基本操作。

15.3.1　启动仿真

1．菜单方式

采用菜单命令形式启动仿真过程是相当简单和方便的。用户可以在配置对话框中完成诸如仿真起止时间、微分方程求解器、最大仿真步长等参数的设置，而不需要去记忆相关指令的用法，尤其是可以在不停止仿真过程的情况下实时地完成以下操作：

- 修改仿真起止时间、最大仿真步长。
- 改变微分方程求解方法。
- 同时进行另一个模型的仿真。
- 用鼠标单击相应的信号线，在出现的浮动窗口中观看信号的相关信息。
- 在一定范围内修改模块的某些参数。

用菜单方式启动仿真一般分为以下几个步骤：

[1] 设置仿真参数。选择 Simulation→Model Configuration Parameters 命令，弹出仿真配

置对话框。一旦用户单击 Apply 或 Close 按钮，即表示目前的设置生效。关于如何进行相关参数的配置，下面还会具体讲述。

[2] 开始仿真。选择 Simulation→Start 命令即可启动仿真，也可以直接按 Ctrl+T 组合键，还可以在工具栏上直接单击运行按钮，如图 15-13 所示。仿真结束后，计算机将发出声音来提示用户。

由于模型的复杂程度和仿真时间跨度的大小不同，每个模型的实际仿真时间不同，同时仿真时间还受到机器本身性能的影响。用户可以在仿真过程中选择 Simulation→Stop 命令，或者按 Ctrl+T 组合键人为中止模型的仿真。

用户也可以选择 Simulation→Pause 或 Simulation→Continue 命令来暂停或继续仿真过程。如果模型中包含向数据文件或工作区输出结果的模块，或者在仿真配置中进行了相关的设置，则仿真过程结束或暂停后会将结果写入数据文件或工作区中。

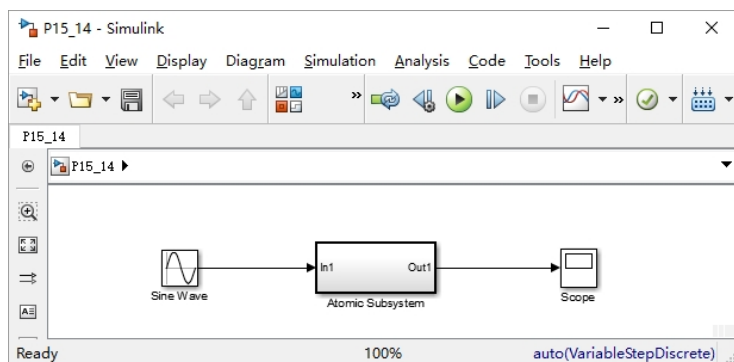


图 15-13　通过菜单命令或工具栏按钮运行仿真

2．命令行方式

相比菜单方式，采用命令行方式启动仿真过程的优点如下：

- 仿真的对象既可以是 Simulink 方框图，也可以是 M 文件或 C-MEX 文件形式的模型。
- 可以在 M 文件中编写仿真指令，从而可以实时改变模块参数和仿真环境。

MATLAB 允许从命令行运行仿真。通过 MATLAB 命令进行模型的仿真使得用户可以从 M 文件来运行仿真，这样就允许不断地改变模块参数并运行仿真，也就让用户可以随机地改变参数来循环地运行仿真，用户就可以进行蒙特卡罗分析了。

MATLAB 提供了 sim 命令运行仿真，该命令完整的语法如下：

```
sim('model', 'ParameterName1',Value1,'ParameterName2', Value2...);
```

其中只有 model 参数是必需的，其他参数都被允许设置为空矩阵（[]）。sim 命令中没有设置的或是设为空矩阵的参数的值等于建立模型时通过模块参数对话框设置的值或是系统默认的值，sim 命令中设置的参数值会覆盖模型建立时设置的参数值。如果仿真的模型是连续系统，那么命令中还必须通过 simset 命令设定 solver 参数，默认的 solver 参数是求解离散模型的 VariableStepDiscrete。

simset 命令用于设定仿真参数和求解器的属性值。该命令的语法如下：

```
simset(proj,'setting1',value1,'setting2',value2,...)
```

由于一般情况下用户不会用到该命令，在此不做详细介绍，如果遇到可以查阅在线帮助，相关的命令还有 simplot、simget、set_param 等。

用 sim 命令仿真上面提到的 Simulink 的演示示例 vdp 方程，代码如下：

```
simOut = sim('vdp','SimulationMode','rapid','AbsTol','1e-5',...
         'SaveState','on','StateSaveName','xoutNew',...
         'SaveOutput','on','OutputSaveName','youtNew');
paramNameValStruct.SimulationMode = 'rapid';
>>paramNameValStruct.AbsTol       = '1e-5';
>>paramNameValStruct.SaveState     = 'on';
>>paramNameValStruct.StateSaveName  = 'xoutNew';
>>paramNameValStruct.SaveOutput     = 'on';
>>paramNameValStruct.OutputSaveName = 'youtNew';
>>simOut = sim('vdp',paramNameValStruct);
mdl = 'vdp';
>>load_system(mdl)
>>simMode = get_param(mdl, 'SimulationMode');
>>set_param(mdl, 'SimulationMode', 'rapid')
>>cs = getActiveConfigSet(mdl);
>>mdl_cs = cs.copy;
>>set_param(mdl_cs,'AbsTol','1e-5',...
       'SaveState','on','StateSaveName','xoutNew',...
       'SaveOutput','on','OutputSaveName','youtNew')
>>simOut = sim(mdl, mdl_cs);
>>set_param(mdl, 'SimulationMode', simMode)
```

3．仿真过程的诊断

如果仿真过程中出现错误，仿真将会自动停止，并弹出一个仿真诊断（Diagnostic Viewer）对话框来显示错误的相关消息，如图 15-14 所示。

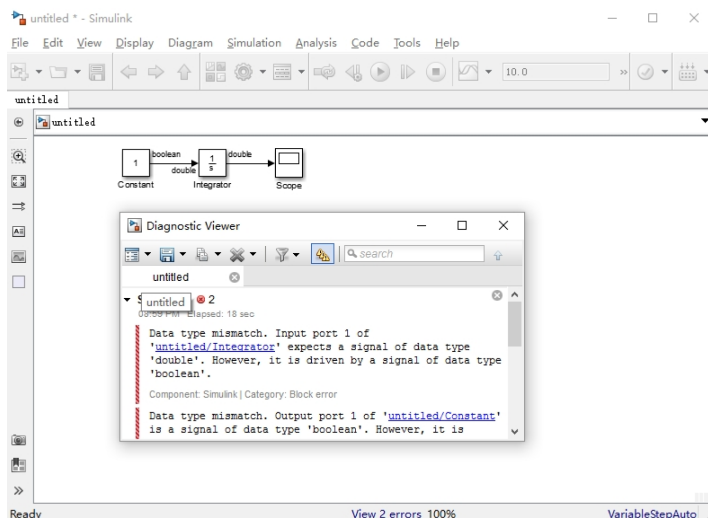


图 15-14 Simulink 的仿真诊断对话框

该对话框分为上下两个部分，上面的部分是每个错误消息的基本情况，下面的部分是对相应错误的详细描述。

仿真诊断对话框将显示的错误信息如下：

- Message：错误类型，诸如模块错误或警告等。
- Source：发生错误的模块名称。
- Fullpath：导致错误的对象的完整路径。
- Summary：错误的简单说明。
- Reported by：报告错误的组件，是 Simulink、Stateflow 还是 Real-Time Workshop 检测到该错误。

Simulink 除了弹出诊断对话框显示错误信息，必要时还会弹出模型方框图，并高亮显示引发该项错误的相关模块。用户也可通过在诊断对话框中双击相关的错误或单击 Open 按钮来弹出相应的方框图。

15.3.2　仿真的配置

1．设置仿真参数和选择求解器

可以通过模型窗口的 Simulation→Model Configuration Parameters 命令打开设置仿真参数的对话框，也可以通过上下文菜单的 Model Configuration Parameters 命令（用鼠标右键单击模型窗口中的空白处即可）打开，如图 15-15 所示。

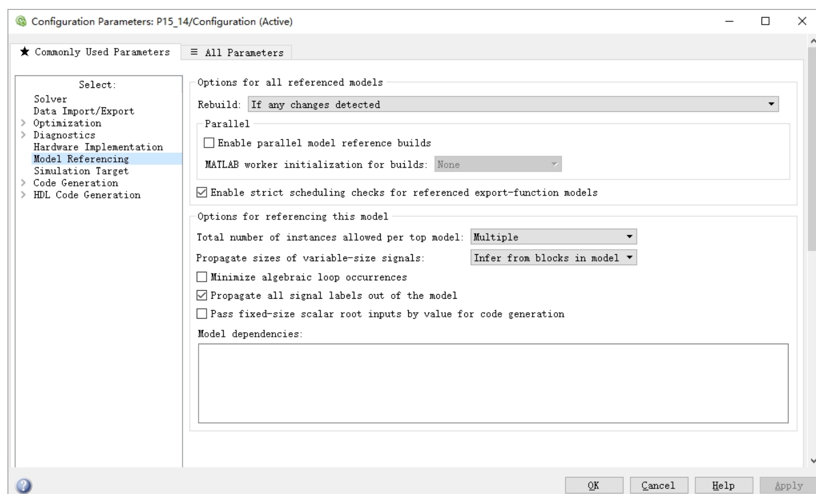


图 15-15　仿真参数对话框

对话框将参数分成不同类型，下面介绍其中 6 组类型，并且对每一组中各个参数的作用和设置方法进行简单的介绍。

（1）Solver。

Solver 面板主要用于设置仿真开始和结束时间，选择求解器，并设置它的相关参数，如图 15-16 所示。

用户可以修改仿真的开始和结束时间，默认仿真从 0.0 秒开始，在 10.0 秒处结束。这里仿真的时间与实际的时钟时间是不同的，后者是仿真计算所花费的实际时间，它的长短与许

多因素有关，如模型的复杂程度、仿真的最大步长及计算机的时钟频率等。

Simulink 模型的仿真一般需要采用微分方程或微分方程组的数值解法。Simulink 为用户提供了各种不同的求解算法，这些算法的使用范围和计算效率各不相同。为了保持仿真的高效性，用户应该根据仿真模型的特点，选择最适合的求解算法。

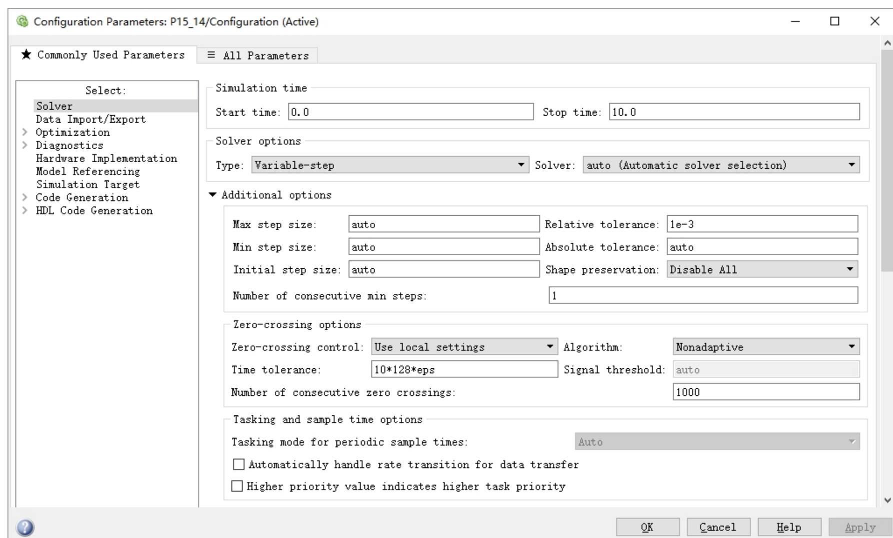


图 15-16　Solver 面板

另外，用户还可以在可变步长或固定步长之间选择。可变步长（Variable-step）提供了误差控制的机制，而固定步长仿真的误差总是采用相同的仿真步骤。

下面将列举 Simulink 提供的不同求解方法。

1）可变步长类。

- ODE45 基于四阶或五阶 Runge-Kutta 算法，属于一步求解法，即计算当前值 *y*(*tn*) 只需要前一步的结果 *y*(*tn*- 1)，是大多数问题的首选求解算法。
- ODE23 基于二阶或三阶 Runge-Kuna 算法，属于一步求解法。在较大的容许误差和中度刚性系统模型下比 ODE45 方法更加有效。
- ODE3 是变阶的 Adams-Bashforth-Moulton PECE 方法，属于多步预测算法，也就是说当前值 *y*(*tn*)需要前几步的结果。
- ODE15s 是基于数值微分公式的变阶算法，属于多步预测算法。如果研究的系统刚度很大，或 ODE45 方法求解的效率不高，可以试试这种算法。
- ODE23s 是基于改进的二阶公式，属于一步求解法。在较大的容许误差情况下，比 ODE15s 更加有效。因此，如果在求解某些刚性系统时，ODE15s 的计算效果不佳，可以改用这种方法。
- ODE23t：一般用来解决中度刚性系统的求解。
- ODE23tb：在较大的容许误差情况下可能比 ODEl5s 方法有效。
- discrete：当系统中没有连续状态变量时选择该方法。

2）固定步长类。

- ODE5 采用固定步长的 ODE45 方法。
- ODE4 属于四阶 Runge-Kutta 方法。
- ODE3 采用固定步长的 ODE23 方法。
- ODE2 采用改进的 Euler 公式。
- ODEl 即 Euler 方法。
- discrete 是固定步长的离散系统求解算法，特别适用于不存在状态变量的系统。

（2）Data Import/Export。

Data Import/Export 面板主要用于向 MATLAB R2018a 工作区输出模型仿真结果数据，或者从 MATLAB R2018a 工作区读入数据到模型，如图 15-17 所示。

- Load from workspace：从 MATLAB R2018a 工作区向模型导入数据，作为输入和系统的初始状态。
- Save to workspace：向 MATLAB R2018a 工作区输出仿真时间、系统状态、系统输出和系统最终状态。

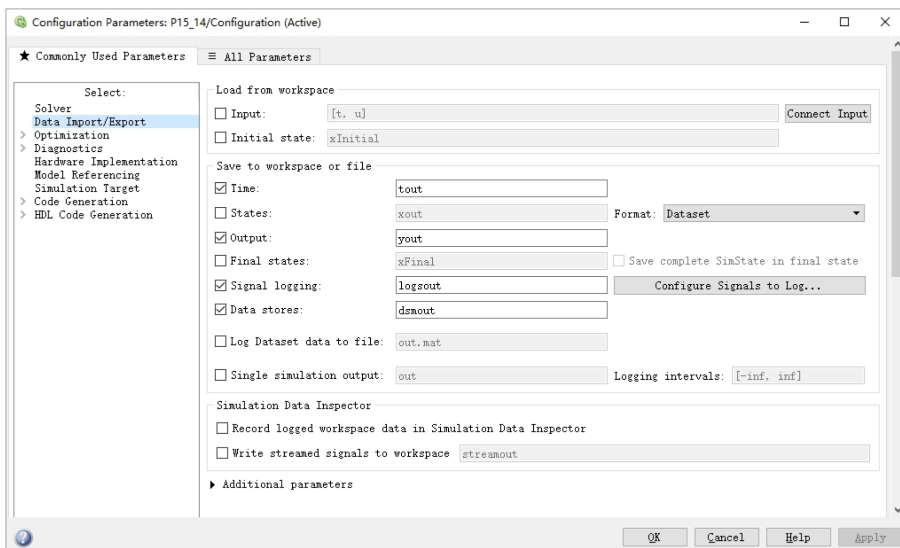


图 15-17 Data Import/Export 面板

- Save options：向 MATLAB R2018a 工作区输出数据的数据格式、数据量、存储数据的变量名及生成附加输出信号数据等。

（3）Optimization。

Optimization 面板用于设置各种选项来提高仿真性能和由模型生成的代码的性能，如图 15-18 所示。

- Block reduction 选项：设置用时钟同步模块来代替一组模块，以加速模型的运行。
- Conditional input branch execution 选项：用于优化模型的仿真和代码的生成。
- Inline parameters 选项：选中该选项使得模型的所有参数在仿真过程中不可调，Simulink 在仿真时就会将那些输出仅决定于模块参数的模块从仿真环中移出，以加快仿真。如果用户要想使某些变量参数可调，那么可以单击 Configure 按钮打开

Model Parameter Configuration 对话框，将这些变量设置为全局变量。

- Implement logic signals as boolean data (vs. double)选项：使得接收布尔值输入的模块只能接收布尔类型。若该项没被选中，则接收布尔输入的模型也能接收 double 类型输入。

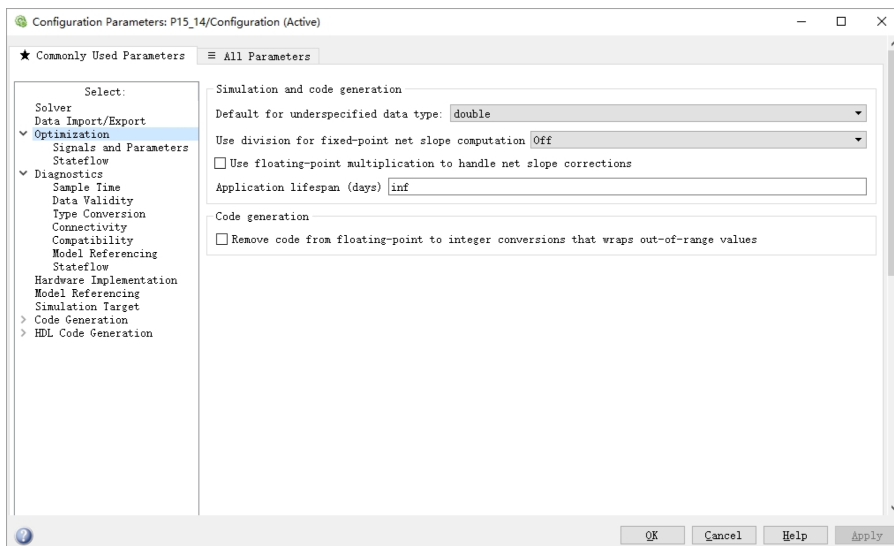


图 15-18　Optimization 面板

（4）Diagnostics。

Diagnostics 面板主要用于设置当模块在编译和仿真遇到突发情况时，Simulink 将采用哪种诊断动作，如图 15-19 所示。该面板还将各种突发情况的出现原因分类列出，各类突发情况的诊断办法的设置在此不做详细介绍。

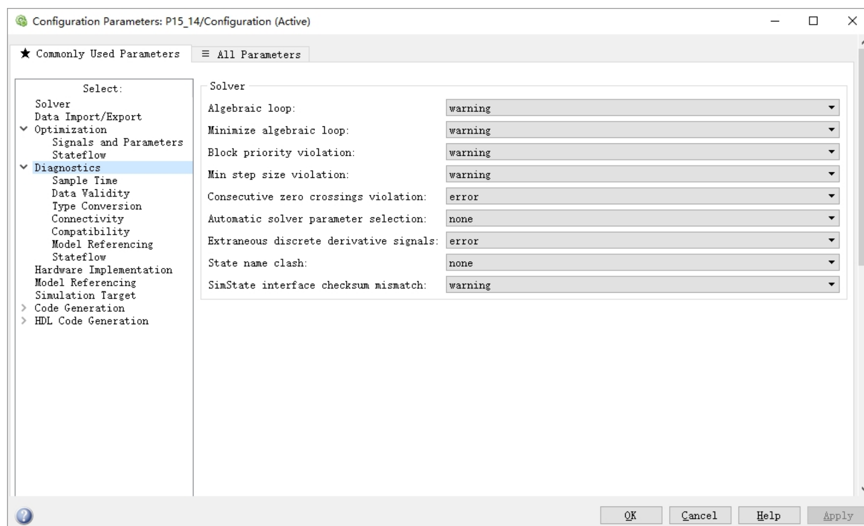


图 15-19　Diagnostics 面板

（5）Hardware Implementation。

Hardware Implementation 面板（如图 15-20 所示）主要用于定义硬件的特性（包括硬件

支持的字长等），这里的硬件是指将来要用来运行模型的物理硬件。这些设置可以帮助用户在模型实际运行目标系统（硬件）之前，通过仿真检测到以后在目标系统上运行可能会出现的问题，如溢出问题等。

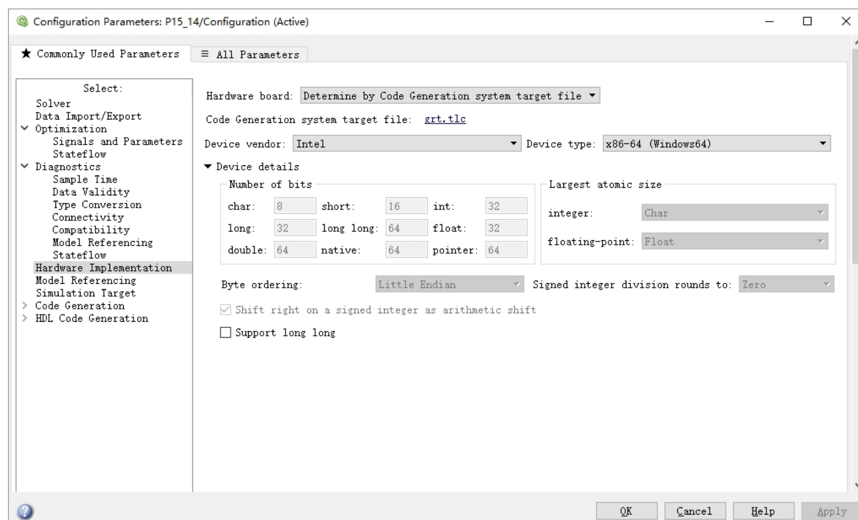


图 15-20　Hardware Implementation 面板

（6）Model Referencing。

Model Referencing 面板主要用于生成目标代码、建立仿真，以及定义当此模型中包含其他模型或其他模型引用该模型时的一些选项参数值，如图 15-21 所示。

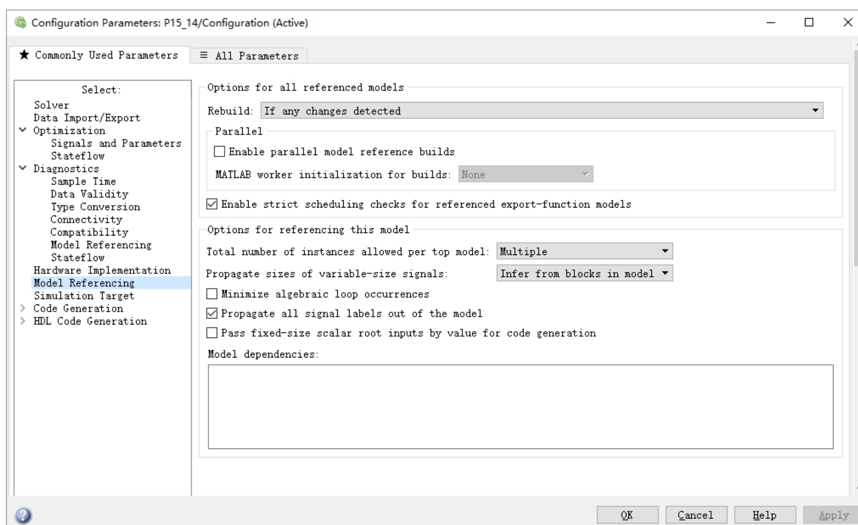


图 15-21　Model Referencing 面板

1）当前模型中含有其他模型时的设置。

- Rebuild：用于设置是否要在当前模型更新、运行仿真和生成代码之前重建仿真和 Real-Time Workshop 目标。因为在进行模型更新、运行仿真和生成代码时，有可能其中所包含的其他模型发生了改变，所以需要在这里进行设置。

2）其他模型中包含当前模型时的设置。

- Total number of instances allowed per top model：用于设置在其他模型中可以引用多少个该模型。
- Model dependencies：用于定义存放初始化模型参数的命令，以及为模型提供数据的文件名（通常是 MAT 文件或 M 文件）或文件的路径。定义的方法是将文件名或文件路径的字符串定义成字符串细胞阵列，如{'E：\Work\parameters.mat', '$MDL\mdlvars.mat', ...'，D:\Work\masks*.m'}。
- Pass fixed-size scalar root inputs by value for Real-Time Workshop：选中此项后，其他模型在调用该模型时就会通过数值来传递该模型的标量输入，否则就通过参考（如输入的地址）来传递输入。选中此项就会允许模型从速度快的寄存器或局部存储单元读取数据，而不是从它的实际输入位置来读取。如果模型的输入在一个时间步内发生改变，那么选中此项就会导致仿真出错。例如，当模型的输入/输出共享同一个局部存储单元（代数环），并且在一个时间步被调用多次时，模型只有当输入是通过参考传递时才会看到标量输入在同一时间步内发生的改变。
- Minimize algebraic loop occurrences：选中此项后，Simulink 将试图消除模型中的一些代数环。

15.3.3　优化仿真过程

Simulinik 仿真质量的好坏受到多方面的影响，但归纳起来，一般仿真模型的好坏与仿真参数设置的适当与否有关。对于模型的创建，具体问题可以具体分析，这里无法进行统一描述，只能从仿真参数的角度来描述影响仿真质量的具体因素。

求解器的默认设置能够满足大多数问题的速度和精度要求，但是有时用户调整某些求解器或仿真配置的参数，可能会获得更好的仿真结果，尤其是当用户了解系统模型的基本特性，并将这种特性提供给求解器时，有可能很大程度上改善仿真的结果。

1．提高仿真速度

下列情况将影响实际的仿真速度。

- 模型当中包含 MATLAB Fcn 模块，这时，由于 Simulink 在每一仿真步中将调用 MATLAB 解释器，仿真速度会明显降低。解决的办法是尽可能采用内建函数（built-in Fco）模块或由基本数学模块搭建代替。
- 模型中包含 M 文件形式的 S 函数，S 函数的执行同样需要调用 MATLAB 解释器。解决的办法是将 S 函数转换成一个子系统或 C-MEX 文件形式的 S 函数。
- 模型当中包含记忆模块，该模块将导致变步长的求解器采用一阶算法来满足仿真时间的要求。
- 最大步长设置得太小，导致仿真速度减慢。尽量采用自动方式。
- 容许误差设置得太小。对于大多数情况，采用其默认值（0.1%）就足够了，过于追求仿真精度而将仿真的容许误差设置得太小，会大大增加仿真的时间。

- 仿真时间跨度太大，可改小一些。
- 对于刚性系统采用了非刚性的求解算法，试着采用 ODE15s 算法。
- 模型中的不同采样周期没有倍乘关系，为了满足所有采样周期的需要，Simulink 可能会采用很少的采样时间，即不同采样周期的最大公约数，从而导致仿真速度的下降。
- 模型中存在代数环。
- 模型中将随机信号进入积分模块中，改进的方法是针对连续系统改用受限白噪声块。

2．提高仿真精度

为了检测仿真的准确性，可以将仿真时间设置在一段合理范围运行仿真，然后将相对容许误差减小到 1e-4（默认值为 1e-3），再次运行仿真，比较前后两次仿真的结果，如果最终的仿真结果没有什么不同，就可以初步确定仿真结果是收敛的。

如果仿真没有反映系统启动时刻的动态特性，可以减小仿真的初始仿真步长，使得仿真不至于跳过系统的某些关键特性。

如果得到的仿真结果不稳定，则可能有以下情况：

- 系统本身是不稳定的。
- 使用了 ODEl5s 的求解方法。

如果仿真结果看起来还没有达到所需要的精度，则可以根据情况采取下列措施：

- 对于存在零附近状态的系统，如果设置的绝对容许误差太大，仿真过程可能会在零附近的区域反复进行迭代。解决的方法是减小该值，或在相应的积分模块中单独进行设置。
- 如果减小绝对容许误差不足以提高仿真的精度，可调小相对容许误差来强制减小仿真的步长，从而增加仿真的步数。

3．仿真结果的观察

仿真进行当中，用户一般需要随时绘制仿真结果的曲线，以观察信号的实时变化。在模型当中使用示波器（Scope）是最为简单和常用的方式。

Scope 模块可以在仿真进行的同时显示输出信号曲线。

在 Scope 模块中进行输出显示还不完善，它只显示输出曲线，而没有任何标记。其他的一些 Graph Scope 模块，如 Auto-Scale Graph Scope、Graph Scope、XYGraphScope，除了显示输出曲线，还显示两个轴和彩色的线型，不过它们执行的速度要比 Scope 模块慢得多。

由于示波器模块在仿真中经常用到，下面说说该模块的具体使用方法。

示波器模块可以接收向量信号，在仿真过程中，实时显示信号波形。如果是向量信号，它还可以自动以多种颜色的曲线分别显示向量信号的各个分量。

不论示波器是否已经打开，只要仿真一启动，示波器缓冲区就会接收传递来的信号。该缓冲区数据长度的默认值为 5 000。如果数据长度超过设定值，则最早的历史数据将被冲掉。

在示波器窗口中，3 个图标分别表示 *x-y* 双轴调节、*x* 轴调节和 *y* 轴调节。图标可以根据数据的实际范围自动设置纵坐标的显示范围和刻度。双击图标则打开示波器属性对话框。

在示波器的显示范围内单击鼠标右键，在弹出的快捷菜单中选择 Axes propertes 命令，

弹出纵坐标设置对话框。在相应的输入栏中输入所希望的纵坐标上下限，可以调整示波器实际纵坐标显示的范围。

示波器属性对话框如图 15-22 所示，其各部分的设置如下：

- Main：常规设置，包括输入端口数目等内容。
- Time：仿真时间设置。
- Display：显示窗口内容设置。
- Logging：图形记录内容设置。

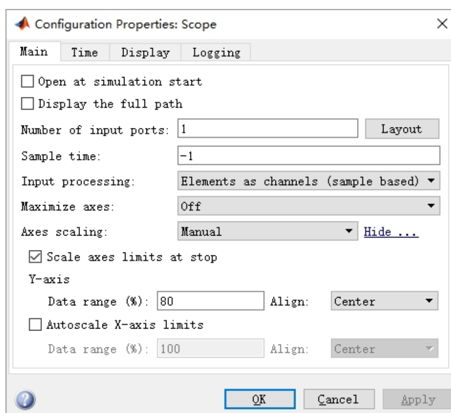


图 15-22　Scope 参数设置

15.4　模型调试

用户创建的仿真模型常常存在这样或那样的错误，即所谓的 Bug。其中有的错误是由于用户在创建模型过程中的疏忽造成的，这些错误在仿真启动前就容易被系统检测出来；但是还有一些错误是因为模型本身的不合理或与原系统存在出入，最终导致仿真结果不正确，甚至影响仿真过程的正常进行。这时只有通过对仿真模型进行一步步的调试，才有可能发现它们并进行修正。

Simulink 提供了强大的模型调试功能，并且有图形界面的支持，使得用户对模型的调试和跟踪更加方便。

本节主要讲述如何利用 Simulink 自身的调试功能对创建的模型进行调试，包括指令方式和图形界面方式，目的是使读者熟悉 Simulink 的调试环境，并初步掌握模型调试的基本操作和方法。

15.4.1　Simulink 调试器

本节通过一个示例简单地向读者介绍如何使用调试器。该示例采用单步运行仿真的方式，用户可以通过这种运行方式来查看在每一个仿真步中各个模块的运行情况，从而检查模块功能是否与用户的设想一致。

选择 Simulation→Debug→Debug Model 命令，打开如图 15-23 所示的调试器。

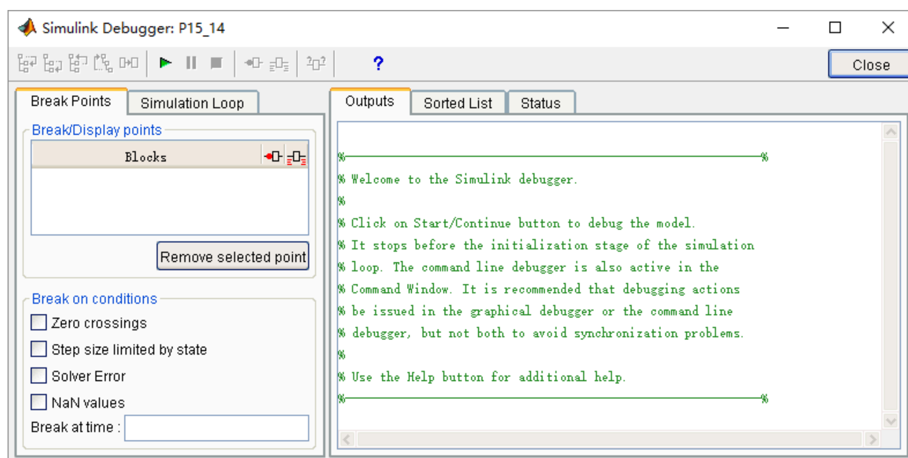


图 15-23　调试器窗口

调试器窗口界面中各项设置的含义如下：

- Break Points 选项卡：用于设置断点，即仿真运行到某个模块方法或满足某个条件就停止。
- Simulation Loop 选项卡：包含 Method、Breakpoints 和 ID 3 项内容，用于显示当前仿真步正在运行的相关信息。
- Outputs 选项卡：用于显示调试结果，包括调试命令提示、当前运行模块的输入/输出和模块的状态。如果采用命令行调试，这些结果在 MATLAB R2018a 命令行窗口中也会显示。调试命令提示显示当前仿真时间、仿真名和方法的索引号。
- Sorted List 选项卡：用于显示被调试的模块列表，该列表按模块执行的顺序排列。
- Status 选项卡：用于显示调试器各种选项设置的值及其他状态信息。

单击▶按钮开始调试，在 Simulation Loop 选项卡中将显示当前运行的方法的名称，并且该方法也将显示在模块窗口中。当调试开始后，窗口会进入调试状态。

15.4.2　命令行调试及设置断点

在命令行调试模式下，用户可以在 MATLAB R2018a 命令行窗口中通过相应的命令来控制调试器。由于命令行调试方式需要用户记住许多命令函数，所以一般用户可以直接通过调试用户界面的相关菜单和工具栏来操作。这里只向读者简单地介绍对于理解命令行调试很关键的几个基本概念。

调试器接收缩写的调试命令，关于这些命令请查阅帮助文件。用户还可以通过在 MATLAB R2018a 命令行按 Enter 键来重复执行前一命令。许多 Simulink 命令和消息是通过 Method ID 和 Block ID 来引用方法和模块的，Method ID 是按方法被调用的顺序从 0 开始分配的一个整数；Block ID 是在仿真的编译阶段分配的，形式为 sid:bid，sid 是用来表示系统的一个整数，bid 是模块在系统中的位置。用户可以通过 slist 命令来查看当前运行模型的每一个模块的 ID。用户启动调试器的两个命令如下：

- sim('vdp',[0,10],simset('debug','on'))。

- sldebug 'vdp'。

其中，vdp 是模型名。

所谓断点就是指仿真运行到此处会停止仿真的地方。当仿真遇到断点停止时，用户可以使用 continue 命令跳过当前断点继续运行到下一个断点。调试器允许用户定义两种断点，即无条件断点和有条件断点。所谓无条件断点是指运行到此处就停止，而有条件断点是指当仿真过程中满足用户定义的条件时才停止仿真。

如果用户知道自己的程序中某一点或当某一条件满足时就会出错，那么设置断点将很有用。

设置无条件断点的 3 种方式：

- 通过调试器的工具栏先在模型窗口中选择要设置断点的模块，然后单击 Run 按钮。可单击 Remove selected point 按钮删除已设置好的断点。
- 通过调试器的 Simulation Loop 选项卡，在该选项卡的 Breakpoints 列中要设置的断点处选择前面的复选框即可。
- 通过在 MATLAB 命令行窗口中运行相关命令。使用 break 和 bafter 命令可以分别在一个方法的前面和后面设置断点，使用 clear 命令可以清除断点。

break 命令的语法代码：

```
break
break m：mid % mid 为方法 ID
break <sid：bid | gcb> [mth] [tid：TID]% sid：bid 为模块 ID，gcb 为当前选中的模块，
   TID 为任务 ID，mth 为方法名，如：Outputs.Major
break <s：sid | gcs> [mth] [tid：TID]   % sid 为系统 ID，gcs 为当前选中的系统
break mdl [mth] [tid：TID]         % mdl 为当前选中的模型
```

○ 注意

如果读者对 Simulink 的调试过程不是很熟悉，可以通过调试器的在线帮助获得相关的使用介绍。方法是：单击调试器窗口中的在线帮助按钮。当然，用户还可以随时按 F1 键来直接获得上下文的帮助信息。

15.5　Simulink 实例应用

本节重点介绍几个常见的 Simulink 仿真实例，请读者深刻体会 Simulink 用来解决实例问题的方便与实效的功能。

1．基于微分方程的 Simulink 建模

例 15-5，已知物块质量 $m=1$kg，阻尼 $b=3$ N.sec/m，弹簧系数 k=90N/m，且物块的初始位移 $x(0)=0.04$ m，其初始速度 $\dot{x}(0)=0.01$m/sec。要求创建该系统的 Simulink 模型，并进行仿真运行。

（1）建立理论数学模型。对于无外力作用的“弹簧-质量-阻尼”系统，据牛顿定律可写出

$$m\ddot{x}+b\dot{x}+kx=0$$

代入具体数值并整理，可得

$$\ddot{x}=-3\dot{x}-90x$$

（2）启动 Simulink 模块，如图 15-24 所示。

（3）开启空白（新建）模型窗口，并建立模型，如图 15-25 所示。

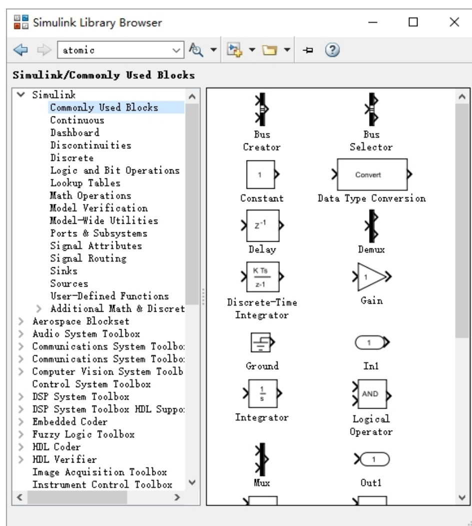


图 15-24　Simulink 模块

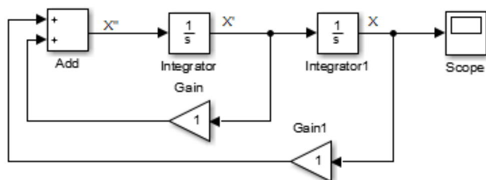
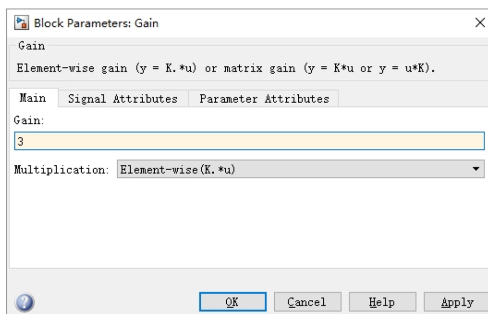


图 15-25　模型框图

（4）设置参数。根据（1）中的数学模型分别设置增益模块的参数值，如图 15-26 所示，并且设置 Add 模块为--。积分模块的参数设置如图 15-27 所示。



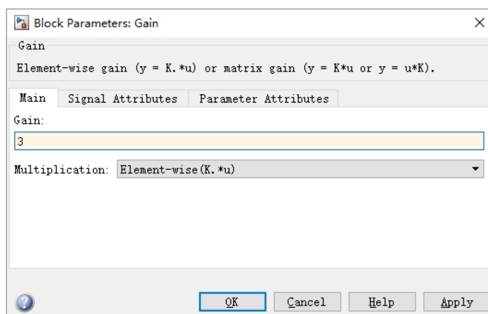
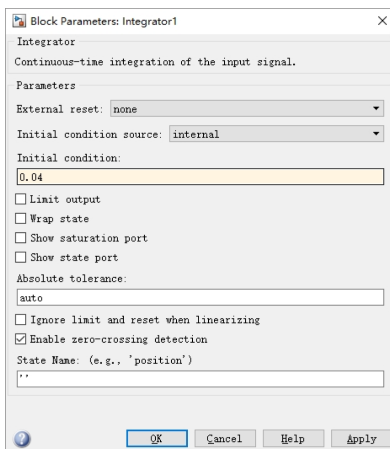


图 15-26　增益参数设置



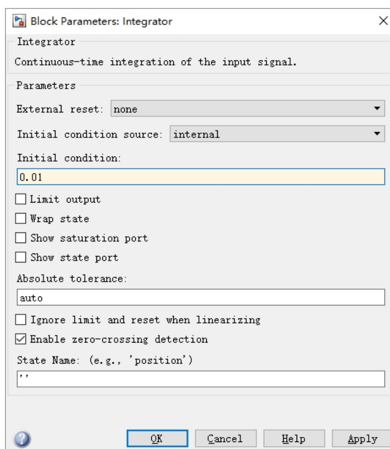


图 15-27　积分参数设置

（5）仿真结果。单击 Run 按钮，单击示波器，出现如图 15-28 所示的图形，同时设置 y 轴范围，以显示完整图像。

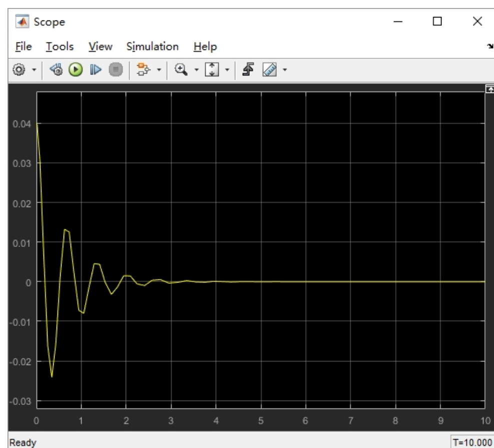


图 15-28　仿真结果

2．离散时间系统的建模与仿真

例 15-6，构建一个低通滤波系统的 Simulink 模型。输入信号是一个受正态噪声干扰的采样信号 $x(kT_s)=2\sin(2\pi kT_s)+2.5\cos(2\pi 10kT_s)+n(kT_s)$，在此 $T_s=0.002$（秒），而 $n(kT)\sim N(0,1)$，$F(z)=\dfrac{B(z)}{A(z)}=\dfrac{1}{1+0.5z^{-1}}$。

（1）建立理论数学模型。

$$y(k)=F(z)x(k)$$

$$F(z)=\frac{B(z)}{A(z)}=\frac{b(1)+b(2)z^{-1}+\cdots+b(n+1)z^{-n}}{1+a(2)z^{-1}+\cdots+a(n+1)z^{-n}}$$

$$F(z)=\frac{B(z)}{A(z)}=\frac{1}{1+0.5z^{-1}}$$

（2）启动 Simulink 模块。

（3）开启空白（新建）模型窗口，并建立模型，如图 15-29 所示。

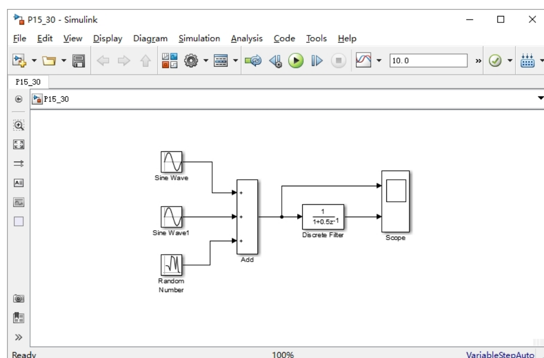


图 15-29　模型图

（4）设置参数。根据题意分别设置信源 Sine Wave 模块的参数值（如图 15-30 所示）、设置随机噪声 Random Number 模块的参数值（如图 15-31 所示），并且设置 Add 模块为+++。Discrete Filiter 模块的参数设置如图 15-32 所示。

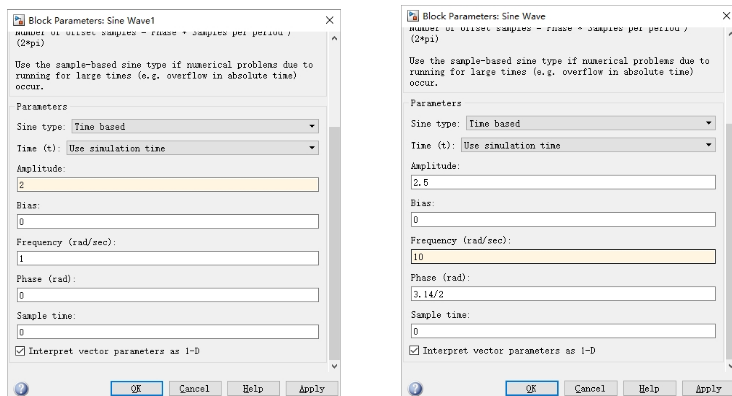


图 15-30　Sine Wave 模块的参数设置

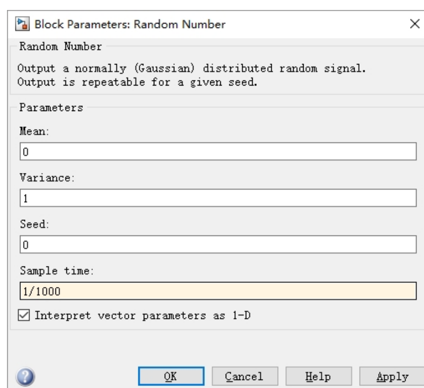


图 15-31 Random Number 模块的参数设置

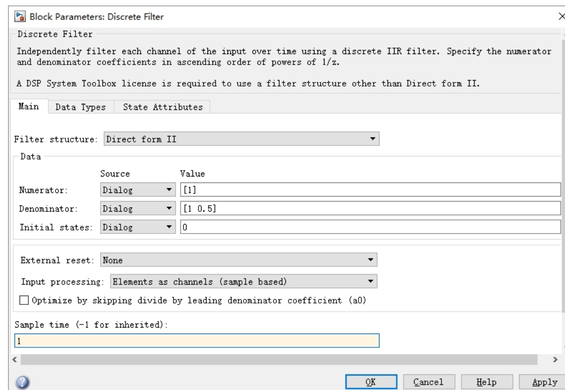


图 15-32　Discrete Filiter 模块的参数设置

（5）仿真结果。单击图 15-29 中的示波器，出现如图 15-33 所示的对话框，并选择独立显示图形，最后单击运行按钮显示如图 15-34 所示的仿真结果。

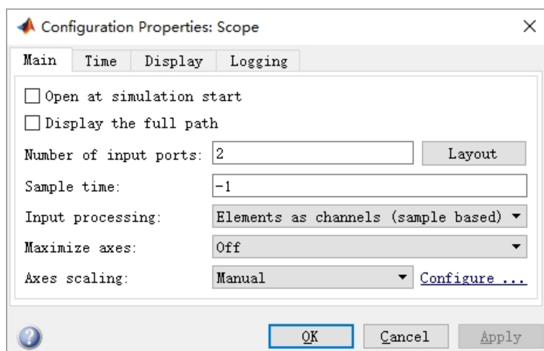


图 15-33　Configuration Parameters:Scope 对话框

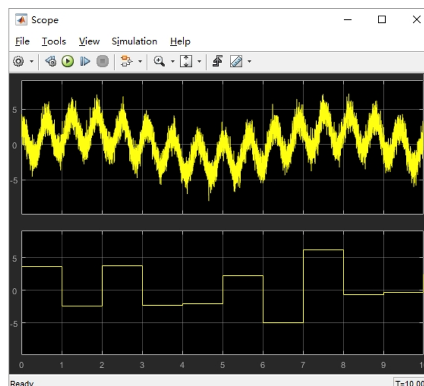


图 15-34　仿真结果

3．调用 MATLAB 工作区中的信号矩阵信源

例 15-7，从 MATLAB 工作区中输入的函数为

$$u(t)=\begin{cases} t & 0\leqslant t<T \\ (4T-t+0.2)^2 & T\leqslant t<2T \\ 1 & else \end{cases}$$

（1）编写一个产生信号矩阵的 M 函数文件，文件名为 source，代码如下：

```
function TU=source(T0,N0,K)
t=linspace(0,K*T0,K*N0+1);
N=length(t);
u1=t(1：(N0+1));
u2=(t((N0+2)：(2*N0+1))-4*T0+0.2).^2;
u3(1：(N-(2*N0+2)+1))=1;
u=[u1,u2,u3];
TU=[t',u'];
```

（2）构造简单的实验模型（如图 15-35 所示）。

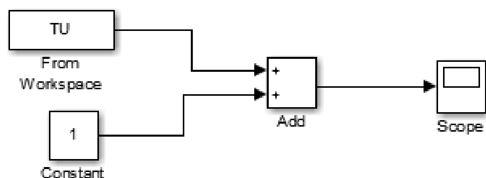


图 15-35　模型图

（3）在指令窗中运行以下指令，会在 MATLAB 工作区中产生 TU 信号矩阵。

```
TU= source (1,50,2);
```

（4）选择 eg1_3 模型窗 Simulation→Start 菜单命令，示波器呈现如图 15-36 所示的信号。

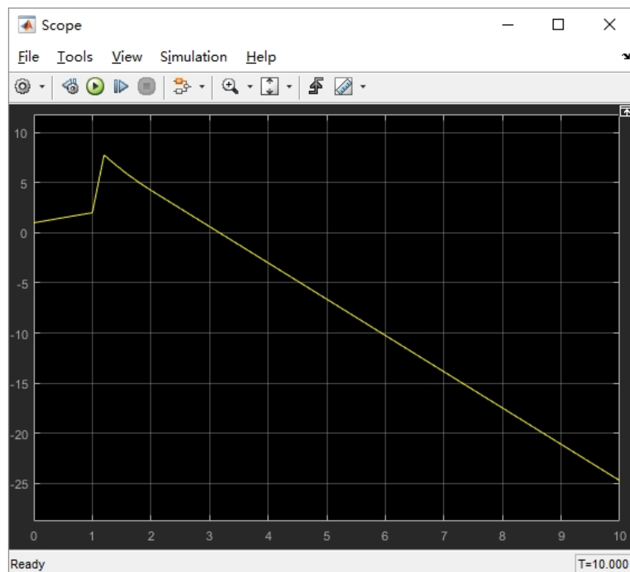


图 15-36　仿真信号

15.6 本章小结

Simulink 是 MATLAB 提供的实现动态系统建模和仿真的一个软件包，它让用户把精力从编程转向模型的构造。Simulink 一个很大的优点是为用户省去了许多重复的代码编写工作。本章针对不同层次读者的需要，较为详尽地介绍了 Simulink 模型创建的步骤和建模技巧、子系统的创建与封装、模型创建后的调试过程。希望读者在学习完本章内容后，能对系统仿真与模型搭建一个比较清晰的认识。

第 4 部分

第 16 章

句柄图形对象



知识要点

图形对象是进行 MATLAB 数据绘图的基本单元，在任何绘制出的图形中，都有一整套完成和配合完成的图形对象，例如绘制的线、坐标轴、图框等。MATLAB 不但提供这些图形对象以完成绘图，还允许用户对这些对象进行定制。在进行所有的操作前，需要用一个唯一的标识符将各个对象区分开来。在 MATLAB 中，这种标识符由句柄来实现，因此图形对象也可以称为句柄图形对象。

学习要求	知识点 \ 学习目标	了解	理解	应用	实践
	句柄图形体系		√		
	句柄图形对象操作			√	√
	句柄图形对象属性设置			√	√
	Figure 对象		√		√
	Plot 对象	√			√
	Group 对象	√			√
	Annotation 对象		√		√

16.1 句柄图形对象体系

句柄图形（Handle Graphics）系统是 MATLAB 中一种面向对象的绘图系统，该系统提供创建计算机图形所必需的各种功能，包括创建线、文字、网格、面及图形用户界面等。在使用 MATLAB 绘图时，句柄图形对象是绘图的基本对象。

16.1.1 句柄图形组织

句柄图形体系中包括不同层次的句柄图形对象，如图 16-1 所示。其中，阴影填充部分为句柄图形体系中几类主要的对象。

在图 16-1 中，Root 为根对象，其代表绘图屏幕；它是所有对象的父对象，换言之，其余对象均为其子对象。在 MATLAB 中，不同的对象也能满足经典面向对象编程语言的继承特性，即子对象在父对象中继承属性。

所有的对象都包含两种类型的属性：一般属性，用来决定对象的显示和保存的数据；方法属性，用来决定在对对象进行操作时调用什么样的函数。

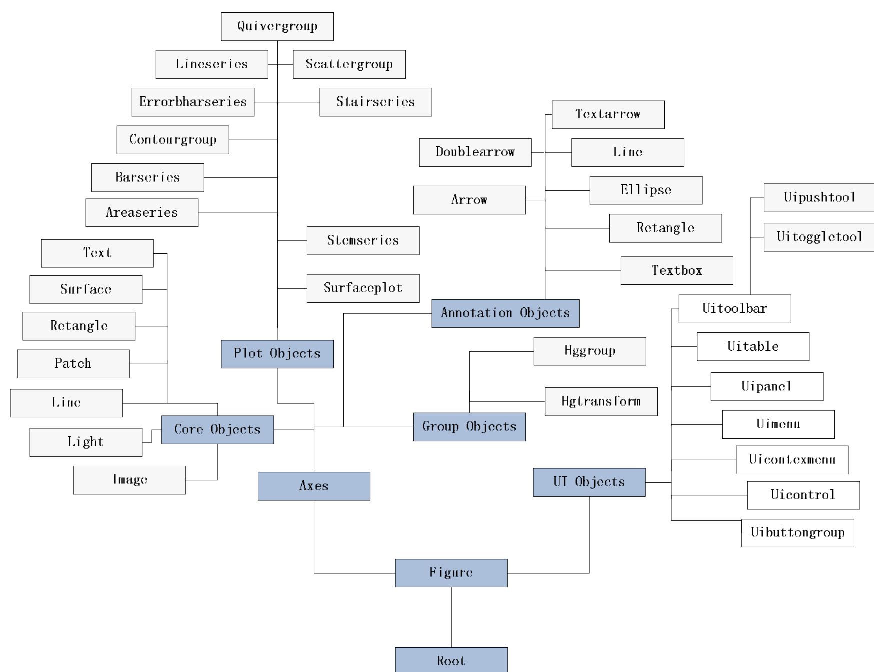


图 16-1　句柄图形体系

16.1.2 句柄图形对象类型简介

从图 16-1 中可以看到，句柄图形对象的类型非常多。这么多的对象，如果任意组织将会给记忆和使用带来极大的不便。幸而，MATLAB 提供了类似如图 16-1 所示的合适组织，

这样就不必记忆所有的对象，而只需要理解其中几个主要变量，便能对句柄图形对象体系进行了解。主要的句柄图形对象及其分类：

（1）核心图形对象（Core Object）：提供高级绘图命令（如 plot 等）及对复合图形对象进行绘图操作的环境。

（2）复合对象：主要包括以下 4 类。

- Plot objects：由基本的图形对象复合而成，提供设置 plot object 属性的功能。
- Annotation objects：同其他图形对象分离，位于单独的绘图层上。
- Group objects：创建在某个方法发挥作用的群对象上。
- UI objects：用于创建用户界面的用户界面对象。

16.2　句柄图形对象操作

句柄图形对象的基本操作包括创建、访问、复制、删除、输出、保存和连接变量等，本节将简要讲述这些操作的实现方法。

16.2.1　创建对象

一幅图形包括多种相关的图形对象，由这些对象共同组成有具体含义的图形或图片。每个类型的图形对象都有一个相对应的创建函数，这个创建函数使用户能够创建该对象的一个实例。对象创建函数名与所创建的对象名相同，例如，surface 函数将创建一个 surface 对象，figure 函数将创建一个 figure 对象。表 16-1 中列出了 MATLAB 所有的对象创建函数。

表 16-1　句柄图形对象创建函数

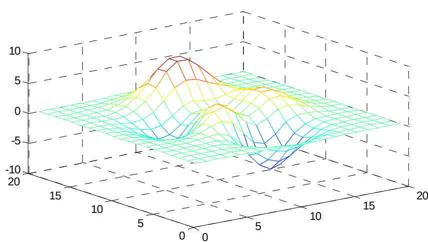
函　数	描　述
axes	在当前图形中创建 axes 对象
figure	显示图形的窗口
hgroup	在坐标轴系统中创建 hgroup 对象
hgtransform	创建 hgtransform 对象
image	创建图像对象
light	创建位于坐标轴中，能够影响面片和曲面的有方向光源
line	创建由顺序连接坐标数据的直线段构成的线条
patch	将矩阵的每一列理解为由一个多边形构成的小面
rectangle	创建矩形或椭圆形的二维填充区域
root	创建 root 对象
surface	创建由矩阵数据定义的矩形创建而成的曲面
text	创建位于坐标轴系统中的字符串
uicontextmenu	创建与其他图形对象相关的用户文本菜单

下面对部分命令的使用进行说明。

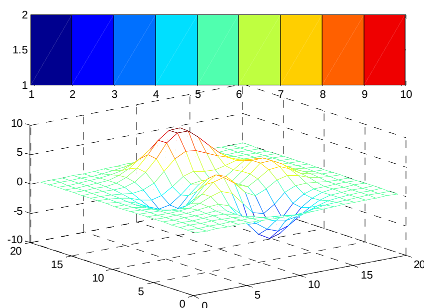
例 16-1，下面的程序在一个图形窗口中创建多个坐标轴（Axes）对象，并在创建过程中指定对象创建的位置。

```
axes('position',[.1  .1  .8  .6])
mesh(peaks(20));                   %参考图 16-2（a）
axes('position',[.1  .7  .8  .2])
pcolor([1:10;1:10]);          %参考图 16-2（b）
```

程序运行结果如图 16-2 所示。其中，图（a）为使用前两条命令进行操作后得到的结果，图（b）为使用所有命令进行操作后得到的结果。从图形中可以看到，坐标轴对象的大小和方向均由命令控制。



（a）创建第一个坐标系统并绘图



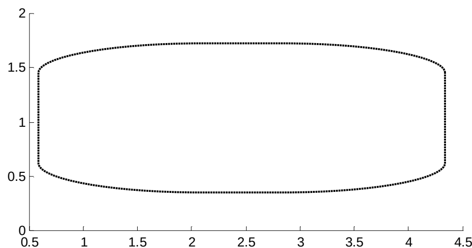
（b）创建第二个坐标系统并绘图

图 16-2　在一个图形窗口中创建多个坐标轴（Axes）对象示例

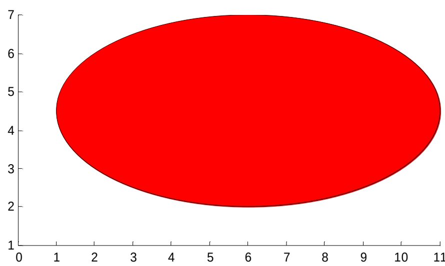
例 16-2，使用 rectangle 命令创建不同的二维矩形区域或椭圆区域示例。程序如下：

```
%以下参考图 16-3（a）
rectangle('Position',[0.59,0.35,3.75,1.37], 'Curvature',[0.8,0.4],...
       'LineWidth',2,'LineStyle','--')
daspect([1,1,1])
%以下参考图 16-3（b）
figure
rectangle('Position',[1,2,10,5],'Curvature',[1,1],...
        'FaceColor','r')
daspect([1,1,1])
xlim([0,11])
ylim([1,7])
```

程序运行结果如图 16-3 所示。



（a）二维矩形区域



（b）椭圆区域

图 16-3　rectangle 命令使用示例

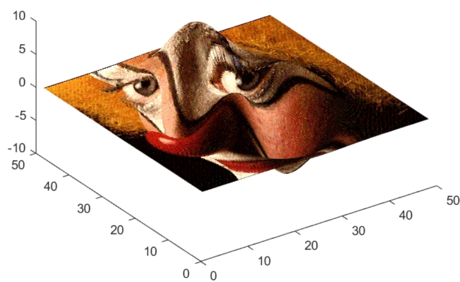
例 16-3，使用 surface 命令将图形映射到面上。程序如下：

```
load clown
surface(peaks,flipud(X),...
  'FaceColor','texturemap',...
  'EdgeColor','none',...
  'CDataMapping','direct')
colormap(map)            %参考图16-4（a）
view(-35,45)             %参考图16-4（b）
```

程序运行结果如图 16-4 所示。



（a）平面视角



（b）立体视角

图 16-4　使用 surface 命令将图形映射到面上

16.2.2　访问对象句柄

MATLAB 创建句柄图形对象时，为每一个句柄图形对象都分配了一个唯一的句柄。访问对象句柄的操作首先是要获取对象的句柄值。有如下两种实现方式：

- 在创建对象时，使用变量获取对象的句柄值。这样，在以后需要使用时，只需要输入相应的句柄值便可以访问对象，并进行后续操作。
- 如果在创建对象时未使用变量获取对象的句柄值，可以使用 findobj 函数，通过特定的属性值来访问句柄图形对象。

前一种方式非常简单，只需要在创建对象时设置相应的变量即可；而后一种方式需要使用 findobj 函数。该函数的调用格式如下：

- h = findobj。
- h = findobj('PropertyName',PropertyValue,...)。
- h = findobj('PropertyName',PropertyValue,'-logicaloperator', PropertyName', PropertyValue,...)。
- h = findobj('-regexp','PropertyName','regexp',...)。
- h = findobj('-property','PropertyName')。
- h = findobj(objhandles,...)。
- h = findobj(objhandles,'-depth',d,...)。

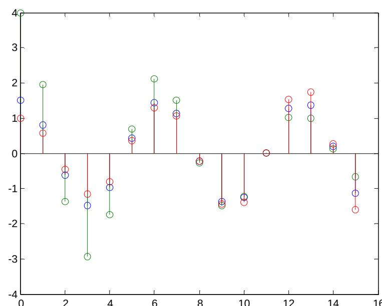
- h = findobj(objhandles,'flat','PropertyName',PropertyValue,...)。

其中，h 为返回的句柄值；PropertyName 为属性名；PropertyValue 为属性值；-logicaloperator 为逻辑运算符，可选-and、-or、-xor、-not，分别代表与、或、异或、非操作；regexp 为正则表达式；-depth 为搜索深度标签；d 为深度值。下面举例说明使用该函数访问句柄图形对象的操作。

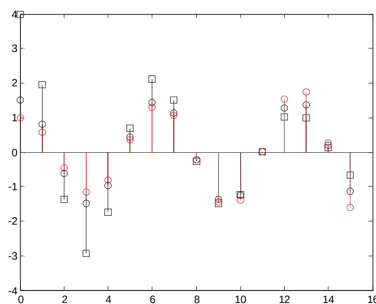
例 16-4，下面的命令为使用 findobj 函数访问对象，并改变对象的属性。

```
x = 0:15;
y = [1.5*cos(x);4*exp(-.1*x).*cos(x);exp(.05*x).*cos(x)]';
h = stem(x,y);
axis([0 16 -4 4])%参考图 16-5（a）
set(h(1),'Color','black',...
        'Marker','o',...
        'Tag','Decaying Exponential')
set(h(2),'Color','black',...
        'Marker','square',...
        'Tag','Growing Exponential')% 参考图 16-5（b）
set(h(3),'Color','black',...
        'Marker','*',...
        'Tag','Steady State')% 参考图 16-5（c）
set(findobj(gca,'-depth',1,'Type','line'),'LineStyle','--')
h = findobj('-regexp','Tag','^(?!Steady State$).');
set(h,{'MarkerSize'},num2cell(cell2mat(get(h,'MarkerSize'))+2))
h = findobj('type','line','Marker','none',...
  '-and','-not','LineStyle','--');
set(h,'Color','red')% 参考图 16-5（d）
```

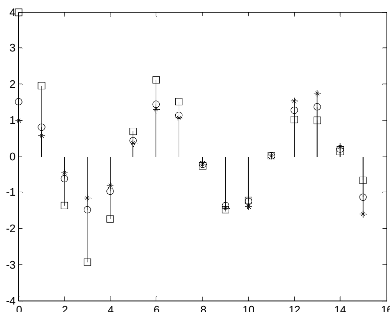
程序运行结果如图 16-5 所示。



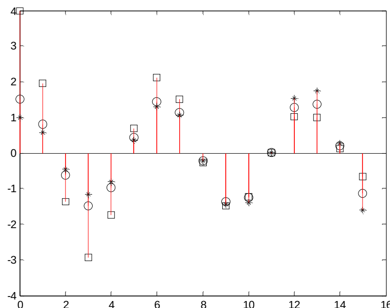
（a）改变 h(1)属性后



（b）改变 h(2)属性后



（c）改变 h(3)属性后



（d）改变查找到的对象属性后

图 16-5　使用 findobj 函数访问对象示例

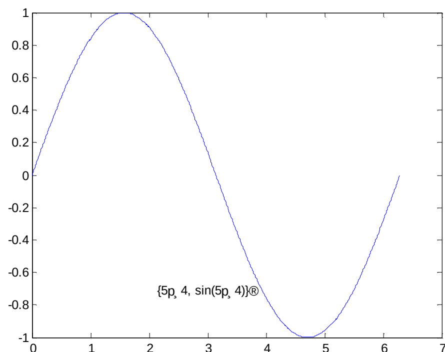
16.2.3　复制和删除对象

在 MATLAB 中可以使用 copyobj 函数复制对象。对于句柄图形而言，可以从一个父对象下复制一个对象，并将其复制到其他父对象中。复制目标和复制结果对象有着同样的属性值，唯一不同的是父对象的句柄值和自身句柄值。

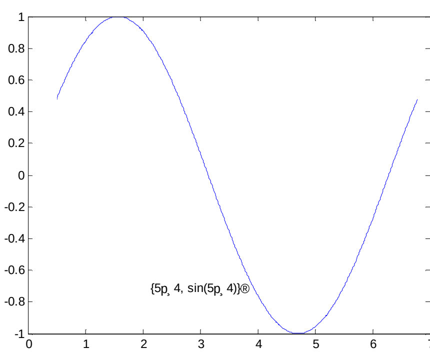
例 16-5，下面的命令在一个绘图窗口中创建一个 text 对象，并将该对象复制到新的绘图窗口中。

```
x=0:0.01:6.28;
y=sin(x);
figure;plot(x,y)
text_handle = text('String','\{5\pi\div4, sin(5\pi\div4)\}\rightarrow',...
    'Position',[5*pi/4,sin(5*pi/4),0],'HorizontalAlignment','right')%参考图 16-6
（a）
x1=0.5:0.01:6.78;
y1=sin(x1);
figure;plot(x1,y1)
copyobj(text_handle,gca)%参考图16-6（b）
```

程序运行结果如图 16-6 所示。其中，图（a）中的文本对象被 copyobj 函数复制到了图（b）中。



（a）待复制的文本对象所在图形



（b）复制生成的文本对象所在图形

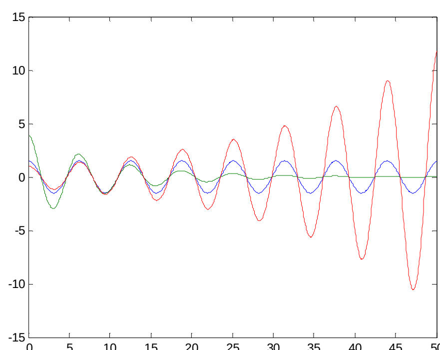
图 16-6　复制生成图形对象示例

使用 delete 函数，可以实现删除对象的操作。

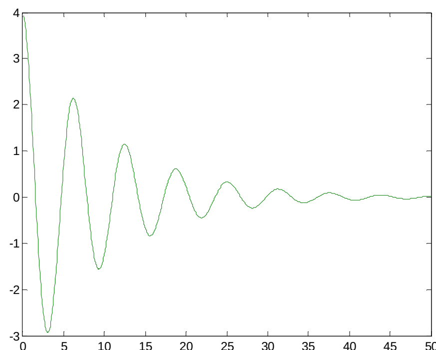
例 16-6，下面的命令首先创建 3 条函数曲线，然后使用 delete 函数删除其中的两条曲线。

```
x = 0:0.05:50;
y = [1.5*cos(x);4*exp(-.1*x).*cos(x);exp(.05*x).*cos(x)]';
h = plot(x,y);
delete(h(1:2:3))
```

程序运行结果如图 16-7 所示。其中，图（a）显示了使用命令创建的 3 条曲线，而图（b）则是删除了其中两条曲线后的图形。



（a）使用命令创建的 3 条曲线



（b）删除两条曲线后的图形

图 16-7　删除句柄图形对象示例

16.2.4　控制图形输出

MATLAB 允许在程序运行过程中打开多个图形窗口，而程序对每个窗口均可进行操作。所以当 MATLAB 程序创建图形窗口来显示图形用户界面并绘制数据时，需要对某些图形窗口进行保护，以免成为图形输出的目标，并且准备好相应的输出窗口以接收新图形。以下内容将讨论如何使用句柄来控制 MATLAB 显示图形的目标和方法，主要包括：

- 设置图形输出目标。
- 保护图形和坐标轴。
- 关闭图形。

1．设置图形输出目标

MATLAB 图形创建函数默认在当前的图形窗口和坐标轴中显示图形，但可以在图形创建函数中通过设置 Parent 属性直接指定图形的输出位置，例如：

```
plot(1:10,'Parent',axes_handle)
```

其中，axes_handle 是目的坐标轴的句柄。在默认情况下，图形输出函数将在当前的图形窗口中显示该图形，且不重置当前窗口的属性。然而，如果图形对象是坐标轴的子对象，为了显示这些图形，将会擦除坐标轴并重置坐标轴的大多数属性。用户可以通过设置图形窗口和坐标轴的 NextPlot 属性来改变默认情况。

MATLAB 高级图形函数在绘制图形前首先检查 NextPlot 属性，然后决定是添加还是重置图形和坐标轴。而低级对象创建函数则不检查 NextPlot 属性，只是简单地在当前窗口和坐标轴中添加新的图形对象。

表 16-2 列出了 NextPlot 属性的取值范围。

表 16-2　NextPlot 属性的取值范围

NextPlot	Figure 对象	Axes 对象
new	创建新图作为当前图形	--
add	添加图形对象	保持不变
replacechildren	删除子对象但并不重置属性	删除子对象但并不重置属性
replace	删除子对象并重置属性	删除子对象并重置属性

hold 命令提供了访问 NextPlot 属性的简便方法。例如以下语句将图形和坐标轴的 NextPlot 属性都设置为 add：

```
hold on
```

而以下语句将图形和坐标轴的 NextPlot 属性都设置为 replace：

```
hold off
```

MATLAB 提供 newplot 函数来简化代码中设置 NextPlot 属性的编写过程。newplot 函数首先检查 NextPlot 属性值，然后根据属性值采取相应的行为。在严谨的操作中，应该在所有调用图形创建函数代码的开头定义 newplot 函数。

当调用 newplot 函数时，可能发生以下动作。

（1）检查当前图形窗口的 NextPlot 属性，具体过程如下：

- 如果不存在当前图形窗口，则创建一个窗口并将该窗口设为当前窗口。
- 如果 NextPlot 值为 add，则将当前窗口设置为当前窗口。
- 如果 NextPlot 值为 replacechildren，则删除窗口的子对象并将该窗口设为当前窗口。
- 如果 NextPlot 值为 replace，则删除窗口的子对象，重置窗口属性为默认值，并将该窗口设置为当前窗口。

（2）检查当前坐标轴的 NextPlot 属性。

- 如果不存在当前坐标轴，则创建一个坐标轴并将其设置为当前坐标轴。
- 如果 NextPlot 值为 add，则将当前坐标轴设置为坐标轴。
- 如果 NextPlot 值为 replacechildren，则删除坐标轴的子对象，并设置当前坐标轴为坐标轴。
- 如果 NextPlot 值为 replace，则删除坐标轴的子对象，重置坐标轴属性为默认值，并设置当前坐标轴为坐标轴。

在默认情况下，图形窗口的 NextPlot 值为 add，坐标轴的 NextPlot 值为 replace。

例 16-7，下面给出一个绘图函数 my_plot 说明 newplot 的使用方法，该函数在绘制多个图形时将循环使用不同的线型。程序如下：

```
function my_plot(x,y)
cax = newplot;  % 返回当前坐标轴对象的句柄
LSO = ['- ';'--';': ';'-.'];
set(cax,'FontName','Times','FontAngle','italic')
set(get(cax,'Parent'),'MenuBar','none')
line_handles = line(x,y,'Color','b');
style = 1;
for i = 1:length(line_handles)
   if style > length(LSO), style = 1;end
   set(line_handles(i),'LineStyle',LSO(style,:))
   style = style + 1;
end
grid on
```

其中，my_plot 函数使用 line 函数绘制数据。虽然 line 函数并不检查图形窗口和坐标轴的 NextPlot 属性值，但是通过调用 newplot 使 my_plot 函数与高级函数 plot 执行相同的操作，即每次用户调用该函数时，函数都对坐标轴进行重置。my_plot 函数使用 newplot 函数返回

的句柄来访问图形窗口和坐标轴。下面的命令调用 my_plot 函数绘图。

```
my_plot(1:10,peaks(10))
```

程序运行结果如图 16-8 所示。

在一些情况下需要改变坐标轴的外观来适应新的图形对象。在改变坐标轴和图形窗口之前，最好先测试一下 hold 属性是否为 on，当 hold 属性为 on 时，坐标轴和图形窗口的 NextPlot 数值均为 add。以下的 my_plot3 函数将接收三维数据并使用 ishold 来检查 hold 属性的状态，以此来决定是否更改视图。

```
function my_plot3(x,y,z)
cax = newplot;
hold_state = ishold;        %检测当前的 hold 状态
LSO = ['- ';'--';': ';'-.'];
if nargin == 2
hlines = line(x,y,'Color','k');
if ~hold_state              %如果 hold 状态为 off，改变视图
view(2)
end
elseif nargin == 3
hlines = line(x,y,z,'Color','k');
if ~hold_state              %如果 hold 状态为 off，改变视图
view(3)
end
end
ls = 1;
for hindex = 1:length(hlines)
if ls > length(LSO),ls = 1;end
set(hlines(hindex),'LineStyle',LSO(ls,:))
ls = ls + 1;
end
```

如果 hold 状态为 on，调用 my_plot3 时将不改变视图。否则有 3 个输入参数， MATLAB 将视图由二维变为三维。

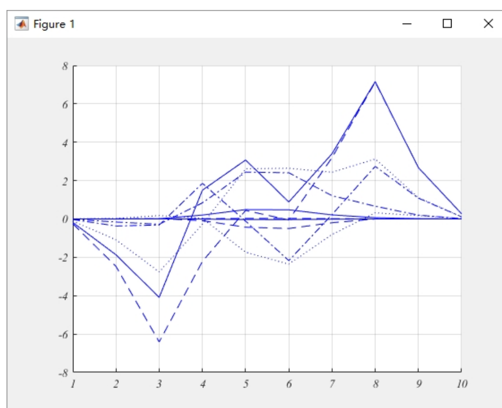


图 16-8 newplot 使用示例

2．保护图形和坐标轴

在进行绘图时，有些情况下需要对图形窗口和坐标轴进行保护以免其成为图形输出的目标。在操作时，可以将特定窗口或坐标轴的句柄从句柄列表中删除，使 newplot 和其他返回

句柄的函数（如 gca、gco、cla、clf、close 和 findobj）无法找到该句柄。这样程序将无法在这个句柄中进行输出，从而保证该图形对象不会成为其他程序的输出目标。

通过 HandleVisibility 和 ShowHiddenHandles 两个属性可以控制保护对象句柄的可见性。HandleVisibility 是所有句柄图形对象都具备的属性，该属性取值如下：

- on：对象句柄可以被任意函数获得，这是该属性的默认值。
- callback：对象句柄对所有在命令行中执行的函数隐藏，而对所有回调函数总是可见的。这种可见程度将保证用户在命令行中输入的命令不会影响被保护对象。
- off：句柄对所有函数，无论是命令行中执行的还是回调的函数，都是隐藏的。

例如，如果一个用户图形界面以文本字符串形式接收用户的输入，并在回调函数中对这个字符串进行处理，那么如果不对窗口进行保护，字符串“close all”有可能会导致用户图形界面的销毁。为了防止这种情况，应将该窗口关键对象的 HandleVisibility 数值暂时设置为 off，相关命令如下：

```
user_input = get(editbox_handle,'String')
set(gui_handles,'HandleVisibility','off')
eval(user_input)
set(gui_handles,'HandleVisibility','commandline')
```

如果被保护的图形窗口是屏幕顶层的窗口，而在它之下存在未保护的窗口，那么使用 gcf 将返回最高层的未被保护的窗口；gca 的情况与 gcf 相同；而如果不存在未保护的窗口或坐标轴，将创建一个窗口并返回它的句柄。

root 对象的 ShowHiddenHandles 属性用于控制句柄图形对象的可见性。ShowHiddenHandles 的默认值为 off；当该属性值为 on 时，句柄对所有函数都是可见的。close 函数可以通过使用 hidden 选项来访问不可见的窗口，命令如下：

```
close('hidden')
```

这时，即使该窗口是被保护的，该语句也将关闭屏幕顶端的窗口。使用以下语句将关闭所有窗口：

```
close('all','hidden')
```

3．关闭图形

发生以下几种情况时，MATLAB 将执行由图形窗口的 CloseRequestFcn 属性定义的回调函数（或称为关闭请求函数）。

- 在图形窗口中调用 close 命令。
- 用户退出 MATLAB 时还存在可见的图形窗口（如果一个窗口的 Visible 属性值为 off，则退出 MATLAB 时并不执行关闭请求函数，而是删除该图形窗口）。
- 使用窗口系统的关闭菜单或按钮来关闭图形窗口。

关闭请求函数有时候会非常有用，它在关闭句柄图形对象时，可以进行如下操作：

- 在关闭动作发生前，弹出提示对话框。
- 在关闭前保存数据。
- 避免一些意外关闭的情况。

默认的关闭请求函数保存在一个名为 closereq 的函数文件中，该函数包括的语句如下：

```
if isempty(gcbf)
  if length(dbstack) == 1
    warning('MATLAB:closereq',...
    'Calling closereq from the command line is now obsolete,...
   use close instead');
  end
  close force
else
  delete(gcbf);
end
```

使用 HandleVisibility 设置关闭请求的图形，在任何没有特殊指明关闭该图形的命令中均可以保护该图形不被关闭，例如：

```
h = figure('HandleVisibility','off')
close         % 图形不关闭
close all     % 图形不关闭
close(h)      % 图形关闭
```

16.2.5　保存句柄

使用图形相关函数时，需要频繁用到句柄来访问属性值和输出图形对象。在一般情况下，MATLAB 提供了一些途径来返回关键对象。然而在函数文件中，这些途径可能并非是获取句柄值的最佳方式，原因如下：

- 在 MATLAB 中查询对象句柄或其他信息的执行效率并不高，在文件中最好还是将句柄值直接保存在一个变量中进行引用。
- 由于当前的坐标轴、图形窗口或对象有可能因为用户的交互而发生变化，查询方式难以确保句柄值完全正确，但是使用句柄变量则可以保证正确地反映对象发生的变化。

为了保存句柄信息，通常在文件开始处保存 MATLAB 状态信息。例如，用户可以使用以下语句作为 M 文件的开头：

```
cax = newplot;
cfig = get(cax,'Parent');
hold_state = ishold;
```

这样，就无须在每次需要这些信息时都重新进行查询。如果对象暂时改变了保持状态，用户应当将 NextPlot 的当前属性值保存下来，以便以后重新设置。

```
ax_nextplot = lower(get(cax,'NextPlot'));
fig_nextplot = lower(get(cfig,'NextPlot'));
…
set(cax,'NextPlot',ax_nextplot)
set(cfig,'NextPlot',fig_nextplot)
```

16.3　对象属性设置

图形对象的属性包括图形对象的外观、行为等很多方面；所谓的属性，包括对象很多方面的信息，如对象类型、子对象和父对象、可视性等。

16.3.1　设置属性

在 MATLAB 中，可以访问任何属性值和设置绝大部分属性值；对一个对象的属性值的设置只会影响该对象，而不会影响其他对象。

在设置属性值时，属性值改变的顺序与命令中对应的属性值关键词出现的先后顺序有关，比如：

```
figure('Position',[1 1 400 300],'Units','inches')
```

该命令首先在指定的位置创建指定大小（默认单位为像素点数）的图形对象；如果改变命令的顺序，则图形的单位为英寸，如：

```
figure('Units','inches','Position',[1 1 400 300])
```

这样，将会产生一个非常大的图形。

常用的设置属性的命令为 set，该命令的调用格式如下：

- set(H,'PropertyName',PropertyValue,...)。
- set(H,a)。
- set(H,pn,pv,...)。
- set(H,pn,MxN_pv)。
- a = set(h)。
- pv = set(h,'PropertyName')。

其中，H 为句柄对象句柄值；PropertyName 为属性名；PropertyValue 为属性值；a 为待设置的句柄值；pn、pv 分别为属性名矩阵和属性值矩阵；MxN 为句柄值矩阵的大小。

在进行设置前，常常需要使用 get 函数对属性进行访问。该函数的调用格式如下：

- get(h)。
- get(h,'PropertyName')。
- <m-by-n value cell array> = get(H,pn) 。
- a = get(h)。
- a = get(0)。
- a = get(0,'Factory')。
- a = get(0,'FactoryObjectTypePropertyName')。
- a = get(h,'Default')。
- a = get(h,'DefaultObjectTypePropertyName')。

其中的参数不再具体讲述，可以参考 set 函数与 MATLAB 帮助文件。下面通过实例对两个函数的使用进行简单说明。

例 16-8，设置属性示例。输入命令：

```
figure('Position',[1 1 400 300],'Units','inches')
set(gcf,'Units','pixels')
get(gcf,'Position')
```

输出结果：

```
%ans =
%    1     1   399   300
```

输入命令：

```
set(gcf,'Units','pixels','Position',[1 1 400 300],'Units',...
   'inches')
get(gcf,'Position')
```

输出结果：

```
%ans =
%    0         0    4.1667    3.1250
```

16.3.2 设置默认属性

设置默认属性是影响所有图形的一个简单方法。在任何时候，只要在新创建的图形中没有定义某个属性值，程序就会采用默认的属性值。因此只需要设置默认的属性，基本上可以影响所有的作图。

下面对默认属性设置的相关内容进行介绍，包括：

- 搜索默认属性。
- 定义默认属性值。

1. 搜索默认属性

MATLAB 对默认值的搜索从当前对象开始，沿着对象的继承关系向上层对象进行搜索，直到 Factory 设置值。

2. 定义默认属性值

设置默认属性值的命令为 set。但与一般的设置使用的属性名不同，一般为在属性名前添加 Default 字样等。例如，利用下面的命令设置线宽：

```
set(gcf,'DefaultLineLineWidth',1.5)
```

命令中的属性名为 DefaultLineLineWidth，而不是 LineWidth。

在设置默认属性后，可以使用命令将图形对象中的属性设置为默认值，操作方法一般为：在设置时，使用 default 作为属性值，如下例所示。

例 16-9，下面的命令首先将 EdgeColor 的默认值设为黑色并进行绘图，再将 EdgeColor 的默认值设为绿色，并将 EdgeColor 设置为默认颜色。

```
set(0,'DefaultSurfaceEdgeColor','k')
h = surface(peaks);            %使用默认 EdgeColor 进行绘图，参考图 16-9
set(gcf,'DefaultSurfaceEdgeColor','g')
set(h,'EdgeColor','default')   %设置为新的默认 EdgeColor 值，参考图 16-10
```

程序运行结果如图 16-9 和图 16-10 所示。

对应设置默认属性，MATLAB 还提供了删除默认属性的方法，其一般方法为将默认属性值设置为 remove，例如：

```
set(0,'DefaultSurfaceEdgeColor','remove')
```

该命令可以删除例 16-9 中设置的默认 EdgeColor 属性值。

还可以将默认属性值设置为 Factory，其方法为在属性后设置属性值为 factory，例如：

```
set(h,'EdgeColor','factory')。
```

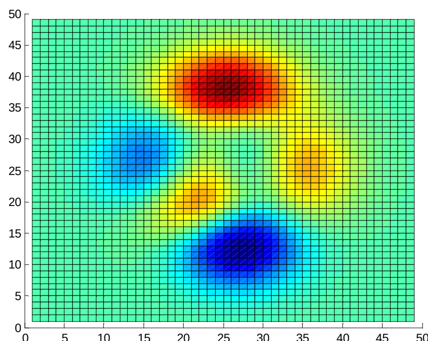


图 16-9　使用默认 EdgeColor 进行绘图

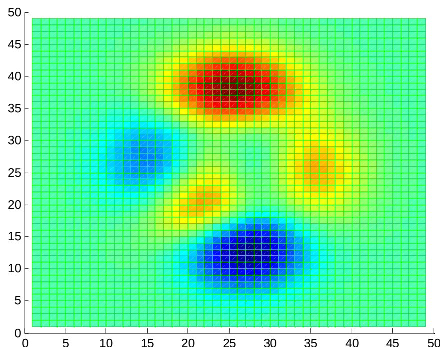


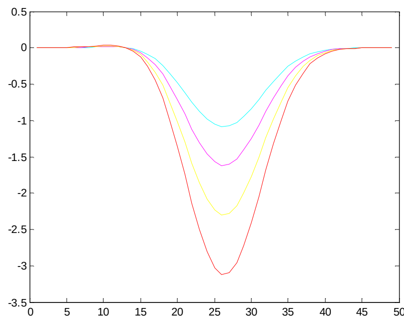
图 16-10　设置为新的默认 EdgeColor 值后

3．设置默认属性

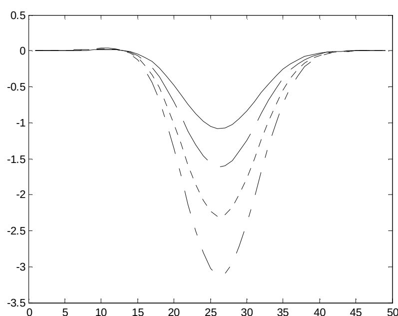
例 16-10，下面的命令分别使用未设置和设置默认线型及颜色进行绘图，并在程序最后删除默认属性值。

```
Z = peaks;
plot(1:49,Z(4:7,:))                         %参考图 16-11（a）
close
set(0,'DefaultAxesColorOrder',[0 0 0],...
     'DefaultAxesLineStyleOrder','-|--|:|-.')
plot(1:49,Z(4:7,:))                         %参考图 16-11（b）
set(0,'DefaultAxesColorOrder','remove',...
     'DefaultAxesLineStyleOrder','remove')
```

程序运行结果如图 16-11 所示。在程序的最后，将这些默认值均进行了删除，以免影响以后使用 MATLAB 进行绘图时得到的线型。



（a）未设置默认属性



（b）设置默认属性后

图 16-11　设置线型默认属性示例

例 16-11，下面的命令设置不同对象层次上对象的默认属性。具体为在同一个绘图窗口中创建两个图形窗口，并在 Figure 对象和 Axes 对象层面使用命令设置默认属性。

```
t = 0:pi/20:2*pi;
s = sin(t);
c = cos(t);
% 设置 Axes 对象的 Color 属性
figh = figure('Position',[30 100 800 350],...
```

```
            'DefaultAxesColor',[.8 .8 .8]);
axh1 = subplot(1,2,1); grid on
% 设置第一个 Axes 对象的 LineStyle 属性
set(axh1,'DefaultLineLineStyle','-.')
line('XData',t,'YData',s)
line('XData',t,'YData',c)
text('Position',[3 .4],'String','Sine')
text('Position',[2 -.3],'String','Cosine',...
    'HorizontalAlignment','right')
axh2 = subplot(1,2,2); grid on
% 设置第二个 Axes 对象的文字旋转属性
set(axh2,'DefaultTextRotation',90)
line('XData',t,'YData',s)
line('XData',t,'YData',c)
text('Position',[3 .4],'String','Sine')
text('Position',[2 -.3],'String','Cosine',...
    'HorizontalAlignment','right')
```

程序运行结果如图 16-12 所示。

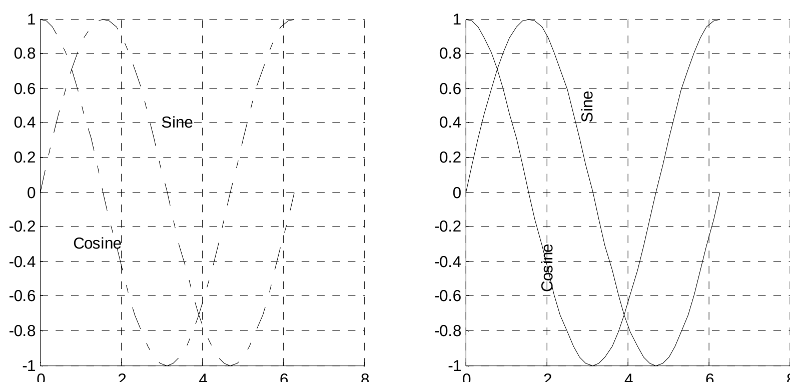


图 16-12　设置不同对象层次上对象的默认属性示例

16.3.3　通用属性

MATLAB 所有的句柄图形对象一般都带有如表 16-3 所示的属性。

表 16-3　句柄图形对象通用属性

属　　性	描　　述	属　　性	描　　述
BeingDeleted	在析构调用时返回一个值	Interruptible	决定回调路径是否可以被中断
BusyAction	控制特定对象的句柄回调函数的中断路径	Parent	父对象句柄值
ButtonDownFcn	控制鼠标动作回调函数路径	Selected	显示对象是否被选取
Children	子对象的句柄值	SelectionHighlight	显示对象被选取状态
Clipping	控制轴对象的显示	Tag	用户定义对象标识
CreateFcn	构造函数回调路径	Type	对象类型
DeleteFcn	析构函数回调路径	UserData	与对象关联的数据
HandleVisibility	控制对象句柄的可用性	Visible	决定对象是否可见
HitTest	决定在鼠标操作时对象是否为当前对象		

16.4　Figure 对象

16.4.1　Figure 对象介绍

在 MATLAB 中，Figure 对象提供图像显示的窗口。该对象的组件包括菜单、工具栏、用户界面对象、坐标轴对象及其子对象，以及其他所有类型的图形对象。

Figure 对象主要提供如下两项功能：

- 包含数据图形。
- 包含图形用户界面（GUI）。

这两项功能虽然可以彼此互相区分，但在操作时也可以同时使用。例如，在一个 GUI 中，也可以绘制数据。

1．用于数据图形

在 MATLAB 中，当不存在 Figure 对象时，使用绘制图形的命令（例如 plot 和 surf 等）可以自动创建 Figure 对象。如果存在多个图形窗口，其中总有一个被设置为当前图形窗口，用于输出图形。可以使用 gcf 相关的命令来获取当前图形窗口的句柄，例如：

```
get(gcf)
```

也可以使用根对象的 CurrentFigure 属性来获取当前 Figure 对象的句柄值；如果没有 Figure 对象将会返回一个空值。如下面的命令：

```
get(0,'CurrentFigure')
```

例 16-12，使用 surf 命令创建 Figure 对象，并进行绘图；在绘图后进行属性设置，使绘制的球面美观。

```
k = 5;
n = 2^k-1;
[x,y,z] = sphere(n);
c = hadamard(2^k);
surf(x,y,z,c);%参考图16-13（a）
colormap([1  1  0; 0  1  1])
axis equal   %参考图16-13（b）
```

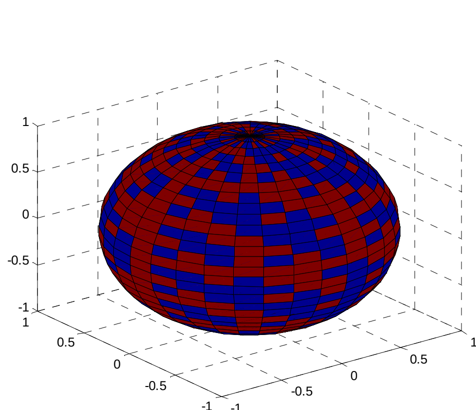
程序运行结果如图 16-13 所示。

2．用于用户图形界面

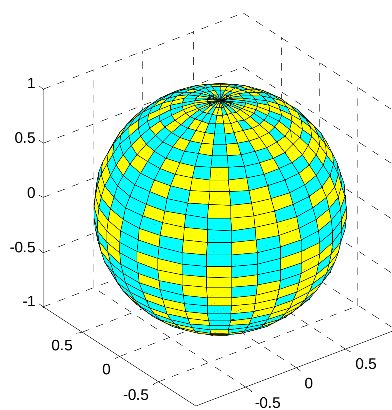
GUI 在交换程序中的使用很普遍，其应用包括从最简单的提示框到极其复杂的交互界面。在使用 Figure 对象去满足 GUI 的需求时，可以对该对象的许多属性进行设置。需要设置的属性如下：

- 显示或隐藏菜单栏（MenuBar）。
- 更改 Figure 对象标识名称（Name）。
- 控制用户对图像句柄的访问（HandleVisibility）。
- 创建回调函数用于用户调整图形时执行其他功能（ResizeFcn）。

- 控制工具栏的显示（Toolbar）。
- 设置快捷菜单（UIContextMenu）。
- 定义鼠标发生动作时的回调函数（WindowButtonDownFcn、WindowButtonMotion- Fcn、WindowButtonUpFcn）。
- 设置图形窗口风格（WindowStyle）。



（a）创建 Figure 对象绘图



（b）更改属性后

图 16-13　surf 命令创建 Figure 对象绘图示例

16.4.2　Figure 对象操作

Figure 对象的操作函数及其简单说明如表 16-4 所示。

表 16-4　Figure 对象的操作函数

函　　数	说　　明	函　　数	说　　明
clf	清除当前图形窗口内容	hgsave	分层保存句柄图像对象
close	关闭图形	newplot	决定绘制图形对象的位置
closereq	默认图形关闭请求函数	opengl	控制 OpenGL 表达
drawnow	更新事件队列与图形窗口	refresh	重新绘制当前图形
gcf	当前图像句柄	saveas	保存图像
hgload	分层加载句柄图像对象	shg	显示最近绘制的图形窗口

16.5　Axes 对象

MATLAB 进行绘图时，需要清楚地知道各个数据点的显示位置，Axes 对象的作用之一即是如此。本节将说明 Axes 对象的一些使用功能：

- 标签与外观。
- 位置。
- 一图多轴。

- 坐标轴控制。
- 线条颜色控制。
- 绘图操作。

16.5.1　标签与外观

MATLAB 程序提供了许多用于控制外观的属性来控制坐标轴的显示，如图 16-14 中显示的一些属性。

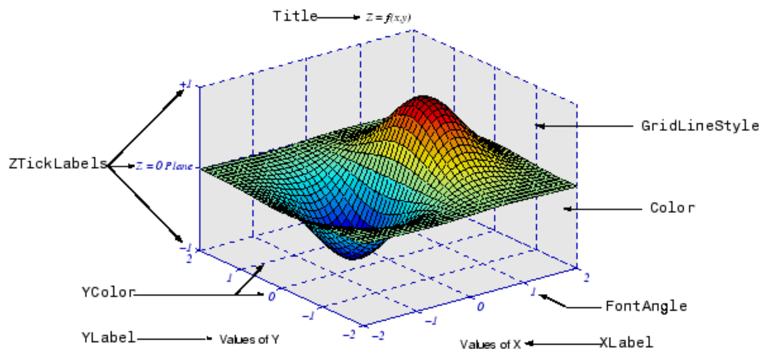


图 16-14　MATLAB 提供的部分坐标轴显示控制属性

在图 16-14 中，其坐标轴属性可以通过下面这条命令设置：

```
h = axes('Color',[.9 .9 .9],...
     'GridLineStyle','--',...
     'ZTickLabel','-1|Z = 0 Plane|+1',...
     'FontName','times',...
     'FontAngle','italic',...
     'FontSize',14,...
     'XColor',[0 0 .7],...
     'YColor',[0 0 .7],...
     'ZColor',[0 0 .7]);
```

可以使用 xlabel、ylabel、zlabel 和 title 等函数来创建坐标轴标签，但需要注意这些坐标轴标签设置函数只对当前坐标轴对象有效。还可以使用 set 函数进行设置，例如下面的命令：

```
set(get(axes_handle,'XLabel'),'String','Values of X')
set(get(axes_handle,'YLabel'),'String','Values of Y')
set(get(axes_handle,'Title'),'String','\fontname{times}\itZ = f(x,y)')
```

对于图形输出要求高的用户，很多时候需要用到字体设置，这时候可以参考下面命令中的设置方式：

```
set(get(h,'XLabel'),'String','Values of X',...
               'FontName','times',...
               'FontAngle','italic',...
               'FontSize',14)
```

16.5.2　位置

坐标轴的位置决定了在图形窗口中坐标轴对象的大小和位置。在 MATLAB 中，坐标轴

位置属性可以使用下面的向量进行表示：

```
[left bottom width height]
```

该向量代表的含义如图 16-15 所示。其中，left 和 bottom 为在二维图形中向量所代表的位置，width 和 height 为在三维图形中向量所代表的位置。

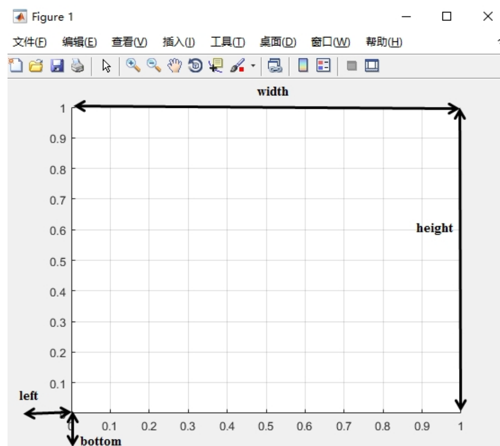


图 16-15　坐标系位置向量说明

在设置位置时，需要注意设置位置使用的单位。在 MATLAB 中可以使用多种单位：

```
set(gca,'Units')
[ inches | centimeters | {normalized} | points | pixels ]
```

其中，{normalized}为归一化的单位，为相对单位，其余均为绝对单位。

16.5.3　一图多轴

在一个图形窗口中创建多个坐标轴对象，最简单也是用得最多的方法是使用 subplot 函数自动计算和设置新的坐标轴对象的位置与大小。然而在更高级的使用中，subplot 函数显然不能满足使用要求。例如，在图形中设置相互重叠的坐标轴对象，以达到创建更有意义的图形的目的。

一图多轴的功能在绘图过程中可以服务的 3 个典型的目的：

- 在坐标轴外放置文本。
- 在同一个图形中显示不同缩放尺度的图形。
- 显示双坐标轴。

1．在坐标轴外放置文本

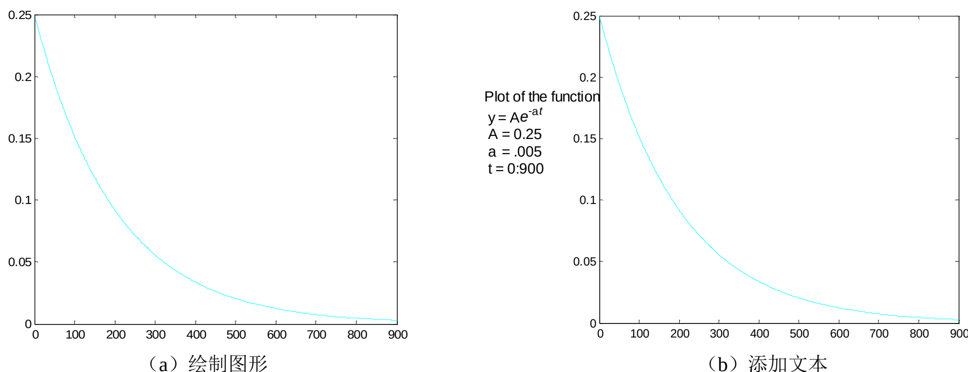
在 MATLAB 中，所有的文本对象均在坐标轴对象的显示范围内，但有时候需要在显示区域外创建文本，这时需要新建坐标轴并在新的坐标轴中创建文本对象，而后再进行显示。

例 16-13，绘制两个坐标轴对象，并使用其中一个对象进行绘图，使用另一个对象放置注解文本。

```
h = axes('Position',[0 0 1 1],'Visible','off');
axes('Position',[.25 .1 .7 .8])
```

```
t = 0:900;
plot(t,0.25*exp(-0.005*t))           %参考图 16-16（a）
str(1) = {'Plot of the function:'};
str(2) = {' y = A{\ite}^{-\alpha{\itt}}'};
str(3) = {'With the values:'};
str(3) = {' A = 0.25'};
str(4) = {' \alpha = .005'};
str(5) = {' t = 0:900'};
set(gcf,'CurrentAxes',h)
text(.025,.6,str,'FontSize',12)      %参考图 16-16（b）
```

程序运行结果如图 16-16 所示。其中，图（a）为放置在当前坐标轴对象中的图形，图（b）为在另一个坐标轴对象中放置文本对象后的图形。



（a）绘制图形　　（b）添加文本

图 16-16　在坐标轴外放置文本示例

2．在同一个图形中显示不同缩放尺度的图形

很多时候，在同一个图形中显示不同缩放尺度的图形将非常有意义，如需要放大图形或整体图形时。下面的例子说明了在同一个坐标轴放置不同缩放尺度图形的操作方法。

例 16-14，在一个图形中放置 5 个不同缩放尺度的球。

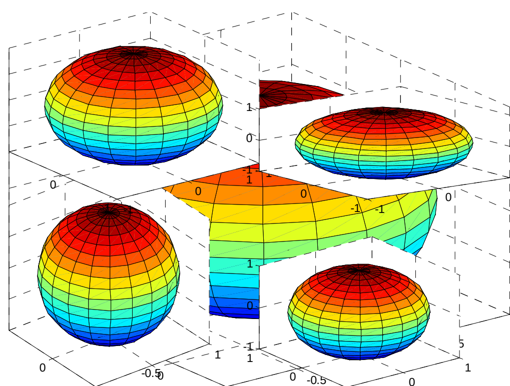
```
h(1) = axes('Position',[0 0 1 1]);
sphere
h(2) = axes('Position',[0 0 .4 .6]);
sphere
h(3) = axes('Position',[0 .5 .5 .5]);
sphere
h(4) = axes('Position',[.5 0 .4 .4]);
sphere
h(5) = axes('Position',[.5 .5 .5 .3]);
sphere                          %参考图 16-17（a）
set(h,'Visible','off')          %参考图 16-17（b）
```

程序运行结果如图 16-17 所示。从图（a）中可以看到每个图都在不同的坐标轴对象中绘制，而从图（b）中看则似乎为在一个坐标轴对象中绘制而成。在同一个图形中显示不同缩放尺度的图形时，消隐不需要的坐标显示很重要。

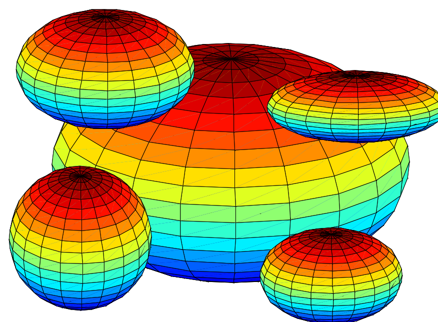
3．显示双坐标轴

使用 XAxisLocation 和 YAxisLocation 属性可以设置坐标轴标签与标度的显示位置，这样就可以在一个图形中创建两个不同的 *xy* 轴显示配对，因为每对显示只需要两个轴便能完

成，而一个图形中有 4 个位置可供显示。这种技术在实际应用中有着较大的价值，其具体的实现方式可以参考例 16-15。



（a）消隐坐标显示前



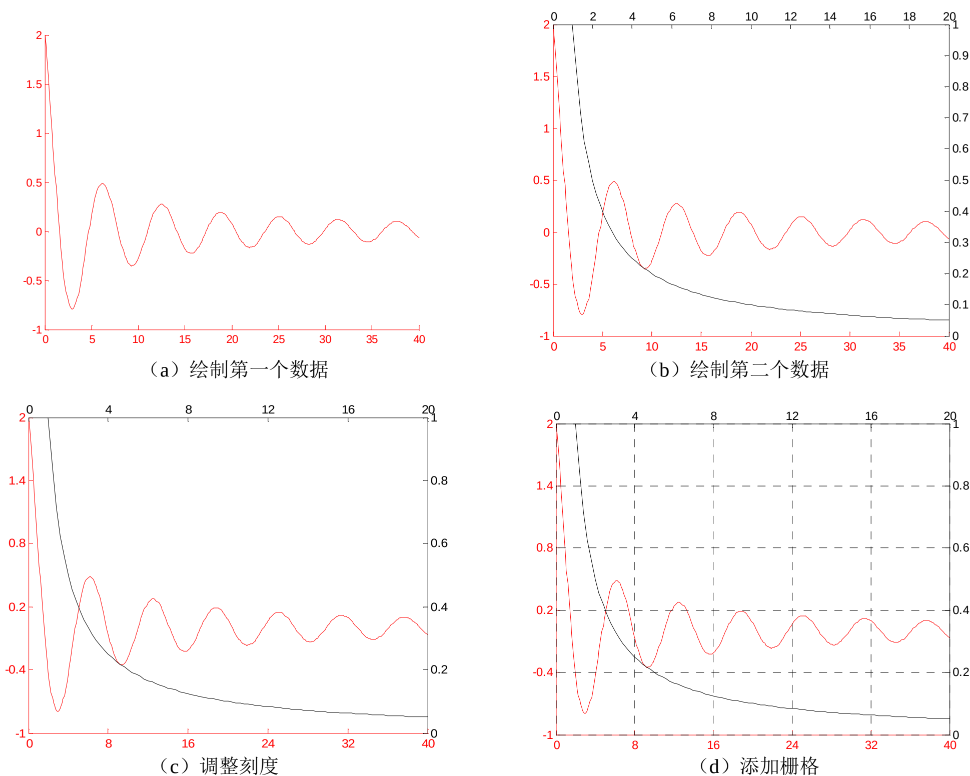
（b）消隐坐标显示后

图 16-17　在同一个图形中显示不同缩放尺度的图形

例 16-15，双坐标轴显示示例。

```
%准备数据
x1 = [0:.1:40];
y1 = 4.*cos(x1)./(x1+2);
x2 = [1:.2:20];
y2 = x2.^2./x2.^3;
%显示第一个坐标轴对象
hl1 = line(x1,y1,'Color','r');
ax1 = gca;
set(ax1,'XColor','r','YColor','r')                %参考图 16-18（a）
%添加第二个坐标轴显示对象
ax2 = axes('Position',get(ax1,'Position'),...
         'XAxisLocation','top',...
         'YAxisLocation','right',...
         'Color','none',...
         'XColor','k','YColor','k');
hl2 = line(x2,y2,'Color','k','Parent',ax2)     %参考图 16-18（b）
xlimits1 = get(ax1,'XLim');
ylimits1 = get(ax1,'YLim');
xinc1 = (xlimits1(2)-xlimits1(1))/5;
yinc1= (ylimits1(2)-ylimits1(1))/5;
xlimits2 = get(ax2,'XLim');
ylimits2 = get(ax2,'YLim');
xinc2 = (xlimits2(2)-xlimits2(1))/5;
yinc2 = (ylimits2(2)-ylimits2(1))/5;
%设置标度显示
set(ax1,'XTick',[xlimits1(1):xinc1:xlimits1(2)],...
      'YTick',[ylimits1(1):yinc1:ylimits1(2)]) %参考图 16-18（c）
set(ax2,'XTick',[xlimits2(1):xinc2:xlimits2(2)],...
      'YTick',[ylimits2(1):yinc2:ylimits2(2)]) %参考图 16-18（c）
%显示栅格
grid on  %参考图 16-18（d）
```

程序运行结果如图 16-18 所示。其基本的绘图步骤为：分别建立坐标轴对象并分别绘制两个数据的图形，然后调整刻度使两者刻度对齐，最后添加栅格增强图形显示的效果。



（a）绘制第一个数据　　（b）绘制第二个数据

（c）调整刻度　　（d）添加栅格

图 16-18　双坐标轴显示示例

16.5.4　坐标轴控制

很多时候，为了得到较好的显示效果，需要对坐标轴的相关属性进行设置，以达到控制坐标轴显示的目的。涉及此目的的相关属性如表 16-5 所示。

表 16-5　坐标轴控制相关属性

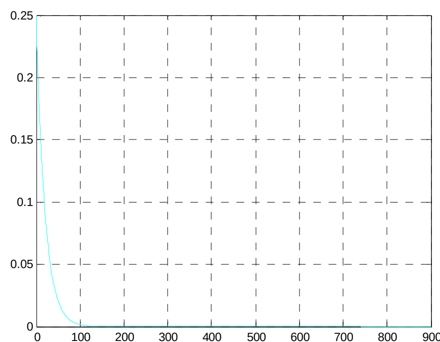
属　性	目　的
XLim、YLim、ZLim	设置坐标轴显示范围
XLimMode、YLimMode、ZLimMode	设置坐标轴显示控制模式
XTick、YTick、ZTick	设置刻度位置
XTickMode、YTickMode、ZTickMode	设置刻度位置控制模式
XTickLabel、YTickLabel、ZTickLabel	设置坐标轴标签
XTickLabelMode YTickLabelMode ZTickLabelMode	设置坐标轴标签控制模式
XDir、YDir、ZDir	设置增量方向

例 16-16，实现坐标轴显示范围控制、标签设置等。

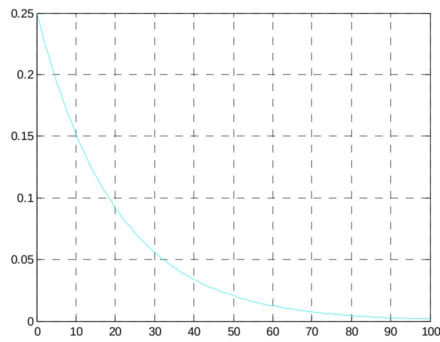
```
%准备数据绘制图形
t = 0:900;
plot(t,0.25*exp(-0.05*t))                          %参考图 16-19（a）
grid on
%调整 x 轴显示范围
set(gca,'XLim',[0 100])                            %参考图 16-19（b）
```

```
%调整 y 轴显示刻度
set(gca,'YTick',[0 0.05 0.075 0.1 0.15 0.2 0.25])  %参考图 16-19（c）
%使用字符串取代刻度值
set(gca,'YTickLabel','0|0.05|Cutoff|0.1|0.15|0.2|0.25') %参考图 16-19（d）
```

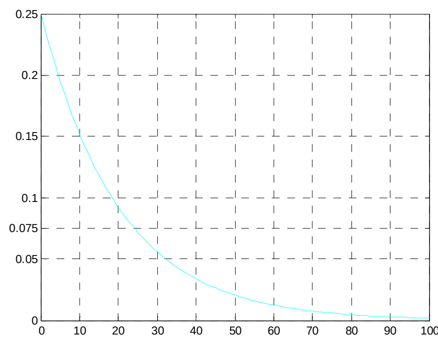
程序运行结果如图 16-19 所示。



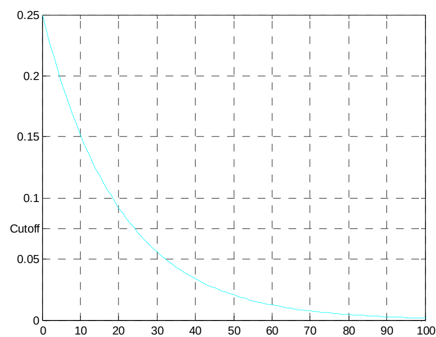
（a）绘制数据



（b）调整 *x* 轴显示范围



（c）调整 *y* 轴显示刻度



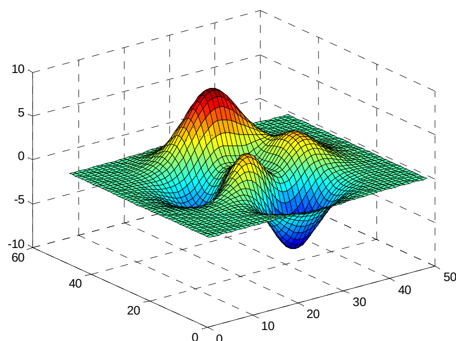
（d）使用字符串取代刻度值

图 16-19　坐标轴显示控制

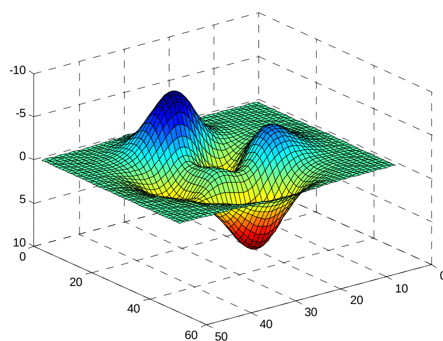
例 16-17，实现坐标轴增量方向的逆转。

```
Z = peaks;
surf(Z)                                          %参考图 16-20（a）
set(gca,'XDir','rev','YDir','rev','ZDir','rev')  %参考图 16-20（b）
```

程序运行结果如图 16-20 所示。图（a）为逆转前的图形，图（b）为逆转后的图形。



（a）坐标增量方向逆转前



（b）坐标增量方向逆转后

图 16-20　坐标增量方向逆转控制

16.5.5　线条颜色控制

控制线条的显示颜色可以获得更好的绘图效果。在坐标轴对象中，与颜色相关的属性如表 16-6 所示。

表 16-6　与颜色相关的属性

属　　性	控制特征	属　　性	控制特征
Color	坐标轴对象的背景颜色	CLim	调色板相关控制
XColor, YColor, ZColor	轴线、刻度、栅格项和标识颜色	CLimMode	调色板相关控制模式
Title	标题颜色	ColorOrder	线颜色自动循环顺序
XLabel, YLabel, Zlabel	标签文本颜色	LineStyleOrder	线风格自动循环顺序

例 16-18，下面的命令将背景颜色设置为白色，并将这个图形使用黑白颜色表示。

```
%设置轴对象中的背景颜色为白色，轴线颜色为黑色
set(gca,'Color','w',...
       'XColor','k',...
       'YColor','k',...
       'ZColor','k')
%设置轴对象中的文本颜色为黑色
set(get(gca,'Title'),'Color','k')
set(get(gca,'XLabel'),'Color','k')
set(get(gca,'YLabel'),'Color','k')
set(get(gca,'ZLabel'),'Color','k'
%设置图形对象的背景颜色为白色
set(gcf,'Color','w')
```

16.5.6　绘图操作

MATLAB 提供的 Axes 对象绘图操作命令如表 16-7 所示。

表 16-7　Axes 对象绘图操作命令

命令函数	操　　作	命令函数	操　　作
axis	设置轴线分度和外观	grid	绘制栅格网线
box	设置坐标轴对象边界	ishold	测试图形保留状态
cla	清除当前坐标轴对象	makehgtform	创建 4×4 变换矩阵
gca	获取当前坐标轴对象句柄值		

例 16-19，使用 grid 命令添加网格线示例。

```
figure
subplot(2,2,1)
plot(rand(1,20))
title('grid off')          %参考图 16-21 左上图
subplot(2,2,2)
plot(rand(1,20))
grid on
title('grid on')           %参考图 16-21 右上图
subplot(2,2,[3 4])
```

```
plot(rand(1,20))
grid(gca,'minor')
title('grid minor')        %参考图16-21下图
```

程序运行结果如图 16-21 所示。

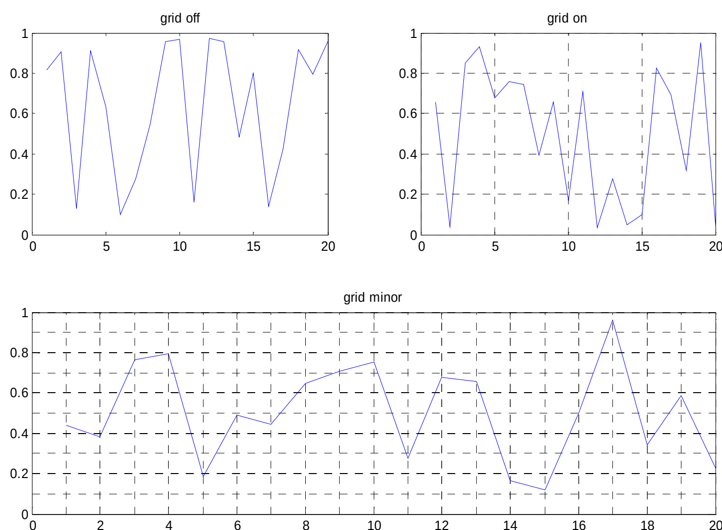


图 16-21　使用 grid 命令添加网格线示例

16.6　Core 对象

16.6.1　Core 对象介绍

Core 对象包括如下基本绘图元素：

- 线、文本、多边形等。
- 特殊对象，如面等。
- 图像。
- 光线对象。

与 Axes 对象不同，Core 对象为绘图元素，而 Axes 更侧重于代表数据。Core 对象的绘图命令如表 16-8 所示。

表 16-8　Core 对象绘图命令

命令函数	操　　作	命令函数	操　　作
axes	创建轴对象	patch	创建斑对象
image	创建图像对象	rectangle	创建矩形对象及椭圆对象
light	创建光线对象	surface	创建面对象
line	创建线对象	text	创建文本对象

图 16-22 说明了一些典型的 Core 图形对象。

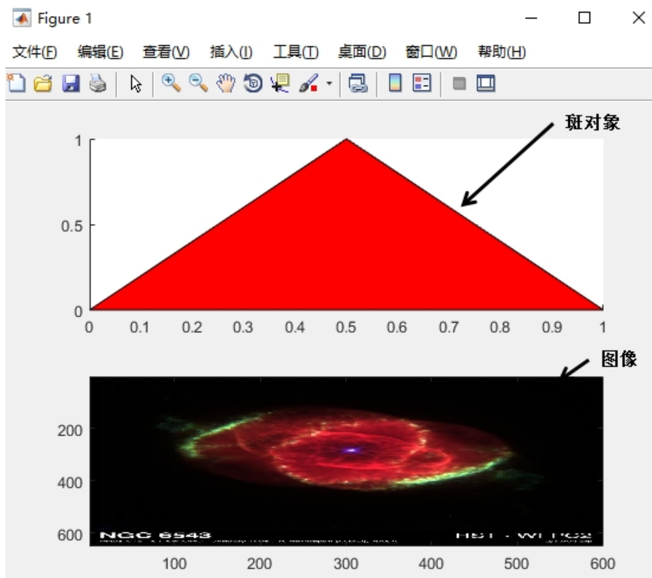


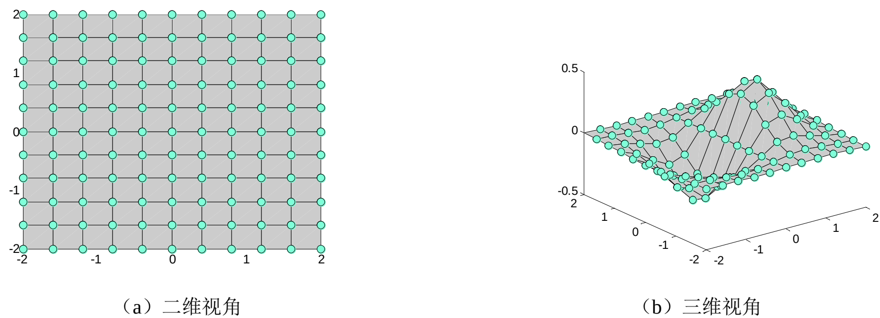
图 16-22　Core 图形对象

16.6.2　Core 对象创建示例

例 16-20，首先对一个数学函数求值，再使用 figure、axes 和 surface 函数创建 3 个图形对象并设置其属性。

```
[x,y] = meshgrid([-2:.4:2]);
Z = x.*exp(-x.^2-y.^2);
fh = figure('Position',[350 275 400 300],'Color','w');
ah = axes('Color',[1 1 1],'XTick',[-2 -1 0 1 2],...
        'YTick',[-2 -1 0 1 2]);
sh = surface('XData',x,'YData',y,'ZData',Z,...
          'FaceColor',get(ah,'Color')-.2,...
          'EdgeColor','k','Marker','o',...
          'MarkerFaceColor',[.5 1 .85]);   %参考图 16-23（a）
view(3)                                    %参考图 16-23（b）
```

程序运行结果如图 16-23 所示。



（a）二维视角　　（b）三维视角

图 16-23　创建 Core 对象示例

16.7　Plot 对象

16.7.1　Plot 对象介绍

在 MATLAB 中，有许多高级绘图函数可以创建 Plot 对象，而且使用这些 Plot 对象的属性值可以简单、快速地访问 Core 对象的重要属性值。Plot 对象可以为 Axes 对象或 Group 对象，如表 16-9 所示。

表 16-9　Plot 对象

对　　象	目　　的	对　　象	目　　的
areaseries	创建 area 图形对象	quivergroup	创建 quiver 或 quiver3 图形对象
barseries	创建 bar 图形对象	scattergroup	创建 scatter 或 scatter3 图形对象
contourgroup	创建 contour 图形对象	stairseries	创建 stairs 图形对象
errorbarseries	创建 errorbar 图形对象	stemseries	创建 stem 或 stem3 图形对象
lineseries	创建 line 图形对象	surfaceplot	创建 surf 或 mesh 群图形对象

16.7.2　Plot 对象创建示例

使用表 16-9 中的对象创建函数均可以创建 Plot 对象，其操作较为简单，下面举例进行说明。

例 16-21，创建一个 contour 对象和一个 surf 对象。

```
[x,y,z] = peaks;
subplot(121)
[c,h] = contour(x,y,z);
set(h,'LineWidth',3,'LineStyle',':')
subplot(122)
surf(x,y,z);
```

程序运行结果如图 16-24 所示，左图为 contour 对象，右图为 surf 对象。

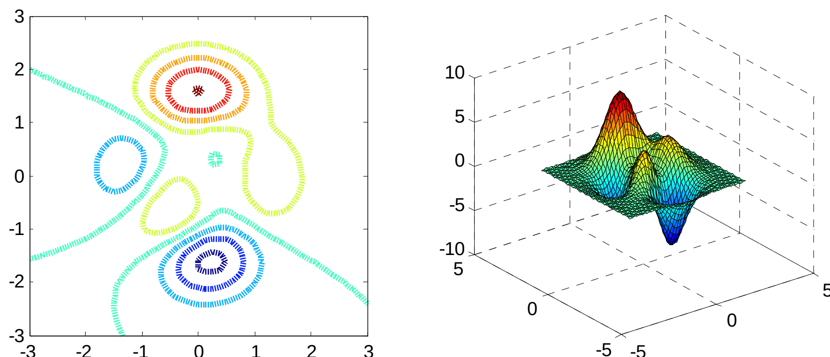


图 16-24　创建 contour 对象和 surf 对象

16.7.3　连接变量

使用 Plot 对象可以连接包含数据的 MATLAB 表达式。例如，lineseries 对象带有 XData、YData 和 ZData 属性的数据来源属性，也被称为 XDataSource、YDataSource 和 ZDataSource 属性。

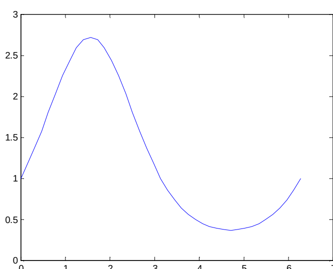
正确地使用数据来源属性，需要注意如下几点：

- 设置数据来源属性的属性值为一个数据变量名。
- 计算变量的最新值。
- 调用 refreshdata 函数更新对象数据。

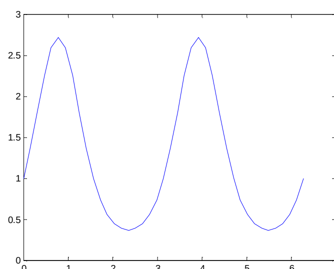
例 16-22，通过连接数据实现所绘图形中数据的自动更新。

```
t = 0:pi/20:2*pi;
y = exp(sin(t));
h = plot(t,y,'YDataSource','y');   %参考图 16-25（a）
for k = 1:2
 y = exp(sin(t.*k));
 refreshdata(h,'caller')        %重新计算 y
 drawnow; pause(.1)
end                                 %参考图 16-25（b）
```

程序运行结果如图 16-25 所示。



（a）原始数据绘图



（b）更新后数据绘图

图 16-25　连接数据示例

16.8　Group 对象

16.8.1　Group 对象介绍

Group 对象提供对由 Axes 子对象构成的对象群进行统一操作的快捷方式。例如，可以设置对象群中所有的对象是否可见、设置可以一次选中所有对象等。Group 对象包括的类型如下：

- hggroup 对象：用于同时创建对象群中的所有对象或控制所有对象的显示等，但在使用前需要用 hggroup 函数先行创建。
- hgtransform 对象：用于同时转换对象群中的所有对象，例如进行旋转、平移和缩放等，但在使用前需要用 hgtransform 函数先行创建。

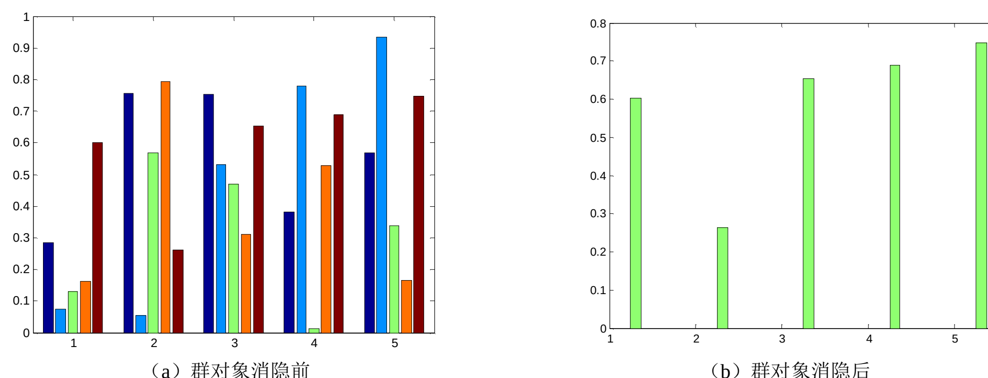
16.8.2 创建 Group 对象

创建 Group 对象的方法很简单，只要将对象群中的对象设置为对象群的子对象即可，而对象群可以为 hggroup 对象或 hgtransform 对象。

例 16-23，创建 hggroup 对象，对对象群进行消隐操作。

```
hb = bar(rand(5));              %创建 5 个 barseries 对象，参考图 16-26（a）
hg = hggroup;
set(hb(1:4),'Parent',hg)        %设置 barseries 对象为 hggroup 对象的子对象
set(hg,'Visible','off')         %消隐对象群中的对象，参考图 16-26（b）
```

程序运行结果如图 16-26 所示。



（a）群对象消隐前　　（b）群对象消隐后

图 16-26　创建 Group 对象示例

16.8.3 对象变换

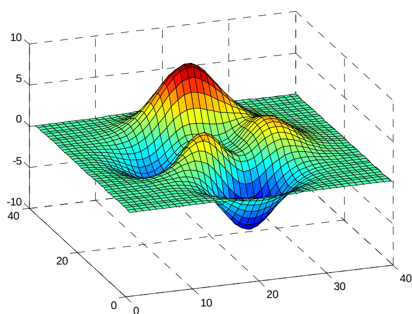
使用 hgtransform 对象可以方便地进行对象变换操作，变换采用的变换矩阵为 4×4 矩阵，可用的变换包括旋转、平移、缩放、透视、倾斜等。关于这些变换的内容在计算机图形学中有详细的叙述和推导，这里不再赘述。

下面使用例子说明使用 hgtransform 对象进行变换的实现方式。

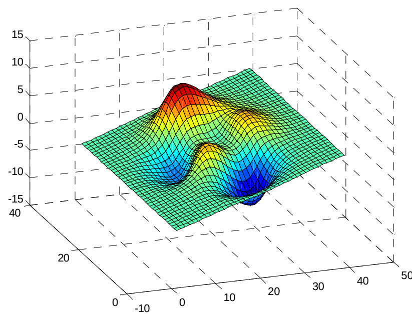
例 16-24，使用 hgtransform 对象进行变换示例。

```
h = surf(peaks(40)); view(-20,30)          %参考图 16-27（a）
t = hgtransform;
set(h,'Parent',t)
ry_angle = -15*pi/180;                     %旋转弧度
Ry = makehgtform('yrotate',ry_angle);      %绕 y 轴旋转矩阵
Tx1 = makehgtform('translate',[-20 0 0]);  %沿 x 轴平移矩阵
Tx2 = makehgtform('translate',[20 0 0]);   %沿 x 轴平移矩阵
set(t,'Matrix',Tx2*Ry*Tx1)                 %参考图 16-27（b）
```

程序运行结果如图 16-27 所示。其中，图（a）为未进行变换的图形，图（b）为变换后的图形。变换的过程为：将图形沿 *x* 轴方向平移-20，绕 *y* 轴旋转-15°，再将图形沿 *x* 轴方向平移 20。



（a）变换前



（b）变换后

图 16-27　对象变换示例

16.9　Annotation 对象

16.9.1　Annotation 对象介绍

Annotation 对象也就是图形中的注释对象，在图形中使用 Annotation 对象可以使图形的组织更为合理，并且更加容易理解。在图形中创建 Annotation 对象可以通过命令方式，也可以通过 GUI 方式。

使用 GUI 方式创建 Annotation 对象的操作非常简单，但需要大量的交互；而使用命令方式创建 Annotation 对象虽然不够直观，但使用起来效率更高，可以节省大量的时间。本节将着重介绍使用命令创建 Annotation 对象的实现方式。

Annotation 对象也是 Axes 对象中的一种，与一般的 Axes 对象不同的是，该对象隐藏了坐标轴。但在使用时，必须清楚地知道，其 Axes 对象的范围为整个图形窗口。

常见的 Annotation 对象包括箭头、双箭头、椭圆、线、矩形、文本箭头、文本框等。这些对象一般在最高层的 Axes 对象中显示。

16.9.2　Annotation 对象使用示例

Annotation 对象的使用方式同其他对象的使用方式区别不大，下面通过简单的例子进行说明。

例 16-25，创建一个注释矩形框，包括一幅图形中的两个子图。

```
%创建图形
x = -2*pi:pi/12:2*pi;
y = x.^2;
subplot(2,2,1:2)
plot(x,y)
h1=subplot(223);
y = x.^4;
plot(x,y)
h2=subplot(224);
```

```
y = x.^5;
plot(x,y)
%计算注释矩形框的位置和大小
p1 = get(h1,'Position');
t1 = get(h1,'TightInset');
p2 = get(h2,'Position');
t2 = get(h2,'TightInset');
x1 = p1(1)-t1(1); y1 = p1(2)-t1(2);
x2 = p2(1)-t2(1); y2 = p2(2)-t2(2);
w = x2-x1+t1(1)+p2(3)+t2(3); h = p2(4)+t2(2)+t2(4);
%创建注释矩形框
annotation('rectangle',[x1,y1,w,h],...
 'FaceAlpha',.2,'FaceColor','red','EdgeColor','red');
```

程序运行结果如图 16-28 所示。

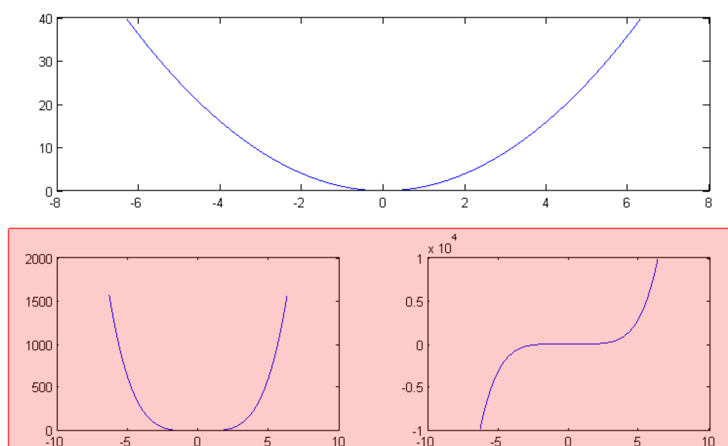


图 16-28　Annotation 对象使用示例

16.10　本章小结

本章介绍了 MATLAB 中句柄图形对象方面的基本知识，包括句柄图形对象体系、句柄图形独享操作、句柄图形对象属性设置等；对句柄图形对象中的六大类型对象，包括 Figure 对象、Axes 对象、Core 对象、Plot 对象、Group 对象和 Annotation 对象的基本知识进行了说明，并举例说明了对这些对象进行操作的基本方式。如果需要了解更多的知识，请参考 MATLAB 帮助文件或其他参考资料。

第 17 章

图形用户界面

知识要点

图形用户界面的设计是 MATLAB 的核心应用之一。当用户与计算机之间或用户与计算机程序之间进行交互操作时，舒服高效的用户接口功能则会对用户产生极大的吸引力。图形用户界面（GUI）则通过窗口、图标、按钮、菜单、文本等图形对象构成用户界面。本章将介绍利用句柄图形用户接口函数实现用户和计算机之间的交互操作过程，如 MATLAB R2018a 的预定义对话框、句柄图形 uicontrol 对象、uimenu 对象、uicontextmenu 对象、uitoolbar 对象等。

学习要求	知识点 \ 学习目标	了解	理解	应用	实践
	用户界面介绍	√			
	用户控件对象				√
	对话框对象			√	√
	菜单对象			√	√
	M 文件编写	√			√
	图形界面创建工具 GUIDE		√		√

17.1　图形用户界面介绍

在 MATLAB R2018a 中，主要通过一系列函数来创建图形用户接口，这些函数主要用来创建用户接口（UI）类型的句柄图形对象。对于这一类函数，在前面章节中已经做过一些介绍。

在 MATLAB R2018a 中，图形用户对象包括了众多的 GUI 对象，其中基本的图形用户界面对象分为 3 类：用户界面控件对象（uicontrol）、下拉式菜单对象（uimenu）和内容式菜单对象（uicontextmenu）。

在图形用户对象中，包括多个图形用户对象及多个图形用户接口容器对象，这些容器对象可以成为图形用户对象或其他容器的接口容器。利用上述对象进行周密的组织、设计，就可以设计出一个界面良好、操作方便、功能强大的图形用户界面。

为了便于读者对这些对象有基本的了解，同时对 GUI 的设计有初步的认识，将这些图形用户界面对象的层次结构进行梳理，绘制出 GUI 图形对象之间的层次树表，如图 17-1 所示。

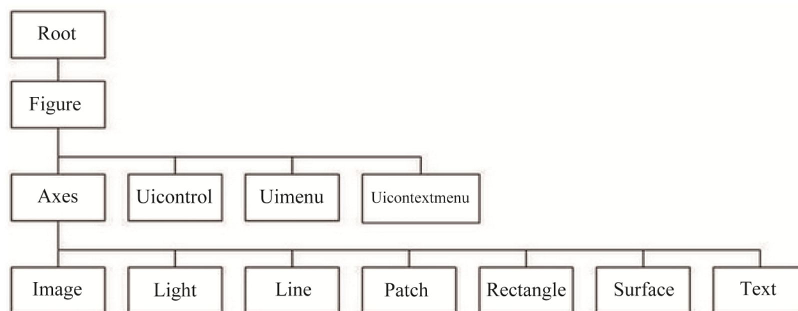


图 17-1　GUI 图形对象之间的层次树表

17.2　图形用户界面控件

在绝大多数的图形用户界面下，都包含控件。控件是图形对象，它与菜单一起用于创建图形用户界面。MATLAB R2018a 中的这些控件对象如表 17-1 所示。通过使用各种类型的控件，可以建立操作简便、功能强大的图形用户界面。

表 17-1　MATLAB R2018a 中常用的控件对象

控件对象类型	说　　明
Push Button	按钮：在按钮上通常显示文本标签。当用户在按钮上单击时，可以使系统 MATLAB R2018a 的回调函数执行某个操作，但不能用于属性或状态的选择。按钮按下后，可以恢复到原来的弹起状态
slider	滑动框：和其他编程语言中的滑动条一样，由滑杆、指示器等部分对象组成。滑动框通常用于从一个数据范围中选择一个数据值。滑动框通常和其他的文本对象一起使用，用来表示滑动框的标题、当前值及滑动框的范围等
Radio Button	单选按钮：通常通过一个文本标签和一个圆圈或菱形构成组件，当选中时，该圆圈或菱形会处于填充状态；而取消选中时，圆圈和菱形则会恢复到无填充状态。单选按钮选择时，通常只能从一组选择对象中选择单个对象
Check Box	复选框：每个复选框对象由复选项和相应的标签对象构成。当复选项选择激活后，复选框中会出现“×”；

（续表）

控件对象类型	说　明
	取消复选后，则会消失
Edit Text	文本编辑框：用户可以动态地修改或替换文本框中的内容。对于文本框对象，用户可以在属性中设置单行或多行文本输入属性。如果设置为单行文本输入，那么用户只能输入一行文本，按 Enter 键后表明输入结束；如果设置为多行文本输入，那么用户可以输入多行文本，换行时需要按 Ctrl+Enter 组合键
Static Text	静态文本：用于显示文本字符串，通常用于显示标题、标签、用户信息和当前值。此时，用户不能对显示的文本进行修改和编辑
Pop-up Menu	弹出式菜单：用户可以从弹出式菜单的多个选项中选择一个选项。当关闭菜单选项时，将会成为一个包含用户选择项的矩形或按钮，位于一定的矩形区域内。有些菜单项中还有其他子菜单选项，同样的方式可以用鼠标完成选择过程
Listbox	列表框：产生的文本条目可以用于选择，但不能进行编辑
Toggle Button	开关按钮：创建切换。和普通按钮不同，开关按钮通常会交替呈现两种不同的状态（弹起或按下）。而普通按钮将会在按下后立即弹起。开关按钮和普通按钮一样，每次单击都会产生相应的操作
Table	表格按钮：创建表格组件。具体使用可参见 uitable 函数
Axes	坐标系：用于在 GUI 中添加图形或图像
Panel	面板：用于将 GUI 中的控件分组管理和显示。使用面板将相关控件分组显示可以使软件更易于理解。面板可以包含各种控件，包括按钮、坐标系及其他面板等。面板包含标题和边框等用户显示面板的属性和边界。面板中的控件与面板之间的位置为相对位置，当移动面板时，这些控件在面板中的位置不改变
Button Group	按钮组：按钮组类似于面板，但是按钮组的控件只包括单选按钮或者开关按钮。按钮中的所有控件，其控制代码必须写在按钮组的 SelectionChangeFcn 响应函数中，而不是用户接口控制响应函数中。按钮组会忽略其中控件的原有属性
ActiveX Control	ActiveX 控件：用于在 GUI 中显示控件。该功能只有在 Windows 操作系统下可用

17.2.1　图形用户界面控件的创建

MATLAB R2018a 提供了命令行和 GUI 设计工具两种方式来创建图形用户界面控件。

1．命令行方式

在命令行方式下，可以通过函数 uicontrol 来建立控件对象。ucontrol 函数有如下几种调用格式：

- Handle=uicontrol(parent)。
- Handle=uicontrol(…, 'PropertyName',PropertyValue,…)。

其中，Handle 是创建的控件对象的句柄值；parent 是控件所在的图形窗口的句柄值；PropertyName 是空间的某个属性的属性名；PropertyValue 是与属性名相对应的属性值。

第一种调用方式，采用 uicontrol 的 Style 属性的默认属性值，在图形窗口的左下角创建一个命令按钮。第二种调用方式，省略控件所在图形窗口的句柄值，表示在当前图形窗口创建控件对象。如果此时无图形窗口，MATLAB 会自动创建一个图形窗口，然后在其中创建控件对象。

例 17-1，命令行方式 uicontrol 的创建示例。

创建一个命令按钮，其位置在[0.5 0.5 0.2 0.1]，单位是 normalized。当单击该命令按钮时，

按钮的位置会随机变动。代码如下：

```
>>h=uicontrol('style','pushbutton','Units','normalized','Position',
[.5 .5 .2 .1],'String','单击此处','Callback','set(h,''Position'',
[.8*rand .9*rand .2 .1])');
```

运行结果如图 17-2 和图 17-3 所示。

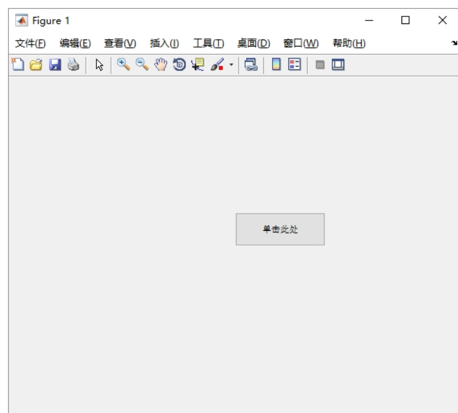


图 17-2 初始运行

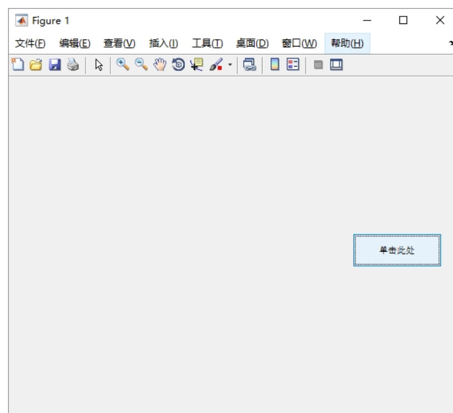


图 17-3 单击后结果

2．GUI 设计工具

在命令行方式下创建控件对象时，需要记住用于创建控件对象的函数。而且要设置控件的属性，必须记住控件的大量属性的属性名，这对于修改控件的属性非常不便。

在 MATLAB R2018a 中利用 GUI 设计工具中的对象设计编辑器（Layout Editor），可以很容易地创建 MATLAB 支持的各种控件，而且通过对象属性检查器（Property Inspector）可以方便地修改、设置创建的控件的属性值。

例 17-2，对象设计编辑器创建控件界面的示例。

设计过程如下：

（1）打开 MATLAB R2018a 工作界面。

（2）在 MATLAB R2018a 工作界面中，选择新建→应用程序→GUIDE 命令，弹出如图 17-4 所示的对话框。

（3）单击“确定”按钮选择默认，此时就会显示对象设计编辑器，如图 17-5 所示。

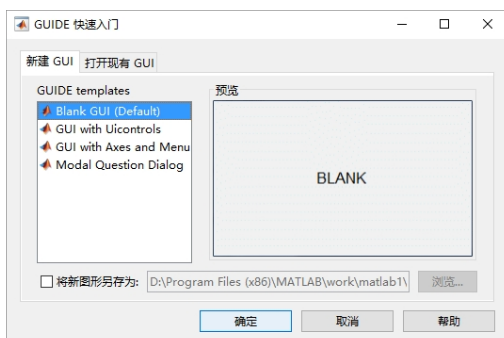


图 17-4 启动界面编辑器

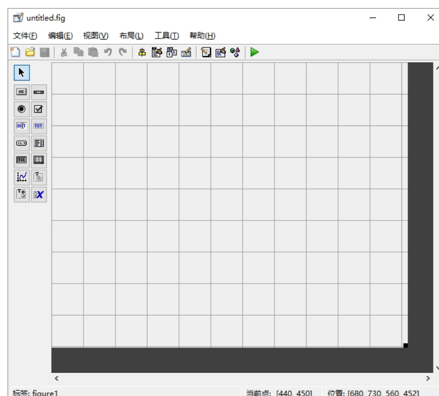


图 17-5 对象设计编辑器界面

接着便可利用该对象设计编辑器创建所需要的控件界面。

（4）在左边选择“面板”控件对象，以拖动方式在对象设计区生成该对象。

（5）同理，创建另外 4 个单选按钮对象，并双击进入各自的检查，修改相应属性如 String 和 ForegroundColor 为自己所需的值。最后保存并命名为 ex17_2。

本例的界面设计运行效果如图 17-6 所示。

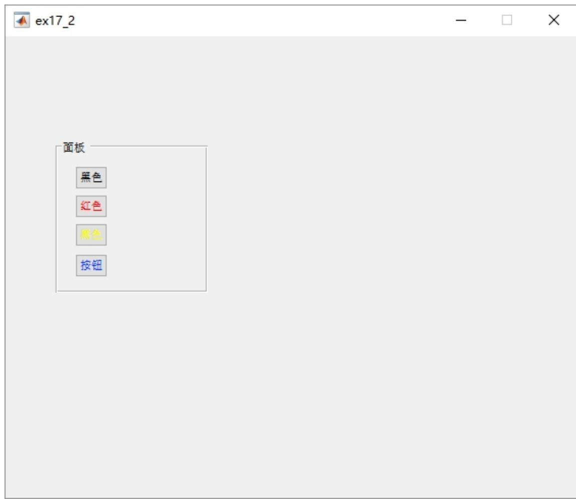


图 17-6　对象设计编辑器创建的控件界面

17.2.2　鼠标动作执行

在 GUI 方式下的应用程序最主要的输入设备就是鼠标，利用鼠标做出判断和交互动作。事实上，如今任何图形界面应用程序与操作系统都提供了对鼠标操作的支持。MATLAB 也提供了对图形用户界面下应用程序鼠标操作的支持。

在 MATLAB R2018a 中，所有的句柄图形对象都拥有一个 ButtonDownFcn 属性，大部分的 UI 对象也都拥有 Callback 属性，对于 uicontrol 对象还拥有一个 KeyPressFcn 属性。图形对象则拥有如 WindowButtonDownFcn、WindowButtonUpFcn、WindowButtonMotionFcn、KeyPressFcn、CloseRequestFcn 及 ResizeFcn 属性。

此外，所有的图形对象还拥有 CreateFcn 和 DeleteFcn 属性。这些属性值都是一个回调字符串，当这些属性激活时，鼠标光标的位置将会决定事件发生时的回调和执行函数。

鼠标光标位于不同的对象时，所能触发该对象的回调函数及响应结果如表 17-2 所示。

表 17-2　鼠标事件的响应

鼠标光标位置及其他条件	响应事件
鼠标光标位于 uimenu 菜单项上，菜单项的 Enable 属性设置为 on	改变 uimenu 对象的外观并为鼠标释放事件做好准备
鼠标光标位于 uicontrol 对象上，且该对象的 Enable 属性设置为 on	改变 uicontrol 对象的外观并为鼠标释放事件做好准备

（续表）

鼠标光标位置及其他条件	响应事件
鼠标光标位于 uimenu 菜单项上，菜单项的 Enable 属性设置为 off	忽略鼠标键按下事件
鼠标光标位于 uicontrol 对象上，该对象的 Enable 属性设置为 off 或 inactive	首先执行 WindowButtonDownFcn 回调函数，然后执行 uicontrol 对象的 ButtonDownFcn 回调代码
鼠标光标位于除 uimenu 和 uicontrol 对象之外的其他图形对象上或附近	首先执行图形对象 WindowButtonDownFcn 回调函数代码，然后执行鼠标所指对象的 ButtonDownFcn 回调函数代码
鼠标光标位于图形对象的内部，但不在其他对象上或附近	首先执行图形对象 WindowButtonDocn 回调函数代码，然后执行该对象的 ButtonDownFcn 回调函数代码

在选择执行回调函数时，MATLAB R2018a 通常会根据图形中的区域来决定回调函数的执行和选择。

（1）在选择时，通常根据 3 个区域来进行选择。

（2）当鼠标光标位于句柄图形对象的 Position 属性确定的范围内时，即可认为鼠标光标位于该对象上。

（3）如果鼠标光标没有位于对象之上，但位于对象的有效选择区域之内，那么可以认为鼠标光标位于该对象附近。

（4）如果鼠标光标既不在一个对象上，也不在一个对象附近，那么鼠标光标远离该对象。

（5）当对象或选择区域出现重叠时，通常会利用对象的叠放次序来确定哪一个对象被鼠标选中。

如果在鼠标光标附近的 uimenu、uicontextmenu、uicontrol 对象的 Enable 属性被设置为 on，那么鼠标单击过程将会触发对象的 Callback 属性字符串的执行。

当按下鼠标时，uicontrol 对象将会做好被触发的准备，通常会改变 uicontrol 对象或 uimenu 的外观，释放鼠标，则会触发相应的回调函数执行过程。如果鼠标光标不在一个 uicontrol 对象或 uimenu 对象上，那么将会按照鼠标按下和鼠标释放过程来执行。

当鼠标释放时，系统将会产生一个鼠标释放事件。当该事件发生时，图形对象的 CurrentPoint 属性首先更新，同时图形对象的 WindowButtonUpFcn 回调函数代码被触发执行。

如果鼠标光标在图形对象的内部移动，将会产生鼠标光标的移动事件。当该事件发生时，图形的 CurrentPoint 属性将会被更新，同时图形的 WindowButtonMotionFcn 回调函数将会被触发执行。

17.2.3　事件队列的执行顺序

创建图形用户界面后，用户可以通过任意方式与该图形用户界面产生交互操作。在进行交互操作的过程中，每次交互都会产生一个事件，这些事件将会形成一系列的事件序列。同时，图形窗口的输入和输出事件也将会作为图形用户界面对象的事件被列入事件队列中。

列入事件队列中的事件包括鼠标光标的移动和触发回调执行的鼠标按键按下或释放操作，以及 waitfor、waitforbuttonpress、drawnow、figure、getframe、pause 等事件。这些事件在需要处理的事件队列中将会按照顺序依次执行和操作。

一般情况下，终止回调函数代码的执行过程是不允许的，但在优先级比较高的命令如 waitfor、drawnow、waitforbuttonpress、getframe、pause 或 figure 等执行时，系统将会把执行过程中的回调函数代码挂起，并检测事件队列中未完成的每个事件。

如果将回调函数相应对象的 Interruptible 属性设置为 on，那么在系统处理完所有未完成事件后，将会恢复被挂起的回调过程，并执行完毕。

如果将该属性设置为 off，那么系统将会只处理没有完成的屏幕刷新事件。

如果对象的 BusyAction 属性被设置为 cancel，那么中断回调执行的事件将会被忽略。

如果将该属性设置为 queue，那么中断回调执行的事件将会被保存在队列中，直到被中断的回调函数执行完毕后再响应该事件。

17.2.4　回调函数的编写

在图形用户界面编写的过程中，使用句柄图形和图形用户界面函数可以充分利用回调函数的代码编写来扩展用户选择的任务执行过程。在执行回调函数的过程中，可以将回调函数设置为回调函数字符串，然后将这些回调字符串传递给 eval 函数来完成执行过程。

例如，对按钮对象执行 buttonfcn click 命令，则可以设置为：

```
H_ uic=uicontrol('Style', 'PushButton', 'Callback', 'buttonfcn click')
```

此时，系统将会把该命令解释为一个在命令行中执行的函数，然后进行函数的执行和操作。因此，可以将需要执行回调的函数都编写在一个回调函数 M 文件中。

当图形用户对象运行执行回调的函数代码时，将会根据该函数来进行函数代码的回调执行过程。如果有多个回调函数需要执行，则可以把这些回调过程编写为 switch 切换结构来完成回调执行过程。

但通过这种方法来编写回调函数代码时，数据的获取和传递将是需要解决的一个重要问题。如可以使用 persistent 声明来限定数据。通过 persistent 函数进行限定时，工作区中所存储的该数据将会在函数执行过程中被使用。但如果有多个图形用户界面对象存在，那么多个图形用户界面对象将共享该 persistent 数据。

当然，传递数据也可以通过句柄图形对象的 tag 属性来进行。

由于句柄图形对象的 tag 属性是句柄图形对象的唯一标识字符串，用户可以使用 findobj 函数来寻找指定的 tag 字符串的图形用户对象。当寻找到该对象后，可以通过 getappdata、setappdata、rmappdata 和 isappdata 函数来对所得到的对象数据进行处理。

这 4 个函数的常用格式如表 17-3 所示。

表 17-3 图形用户对象数据处理函数的格式

数据函数	说 明
Value=getappdata(h,name)	获得应用程序中定义的数值
setappdata(h,name,value)	设置应用程序中的数值
rmappdata(h,name)	删除应用程序中的数值
isappdata(h,name)	判断是否为句柄函数值

例 17-3，回调函数编写示例。

（1）首先新建一个 GUI 文件，如图 17-7 所示。

（2）进入 GUI 开发环境以后添加两个编辑文本框、6 个静态文本框和一个按钮，如图 17-8 所示。设置好各控件以后，就可以为这些控件编写程序来实现两数相加的功能。

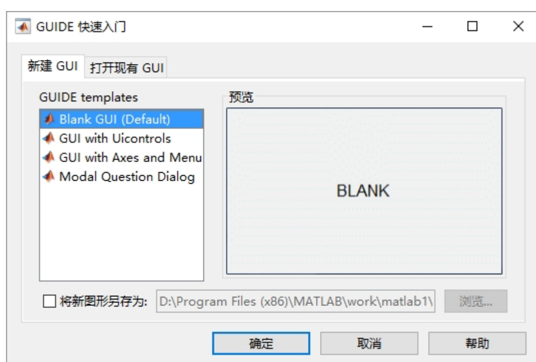


图 17-7 新建一个 GUI 文件

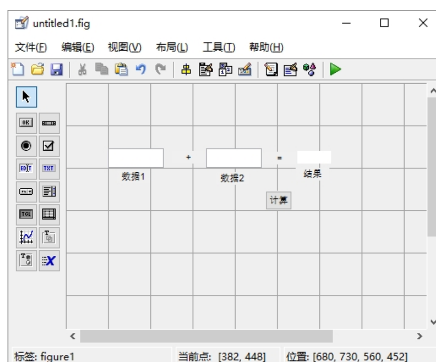


图 17-8 界面设计

（3）先为数据 1 文本框添加代码。

单击编辑器按钮，保存文件。在数据 1 上面的可编辑文本中单击鼠标右键，并选择查看回调→Callback，光标便立刻移到下面这段代码的位置。

```
function edit1_Callback(hObject, eventdata, handles)
% hObject :edit1 对象的句柄（参见 gcbo 用法）
% eventdata:为了兼容 MATLAB 将来版本的保留接口
% handles:拥有句柄的全局变量，可以用来传输用户数据（参见 GUIDATA）
% get(hObject,'String')：返回 edit1 对象的内容为文本形式
% str2double(get(hObject,'String'))：将 edit1 对象的内容转化为双精度型
```

然后在上面这段代码的下面插入如下代码：

```
%以字符串的形式来存储数据文本框 1 的内容。如果字符串不是数字，则显示空白内容
input = str2num(get(hObject,'String'));
%检查输入是否为空。如果为空，则默认显示为 0
if (isempty(input))
set(hObject,'String','0')
end
guidata(hObject, handles);
```

这段代码使得输入被严格限制，我们不能试图输入一个非数字。

（4）同理，为 edit2_Callback 添加同样一段代码。

（5）现在为计算按钮添加代码来实现把数据 1 和数据 2 相加的目的。

在 M 文件中找到 pushbutton1_Callback，代码如下：

```
% --- pushbutton1：在对象上执行单击动作
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject:pushbutton1 对象的句柄（参见 gcbo 用法）
% eventdata:为了兼容 MATLAB 将来版本的保留接口
% handles:拥有句柄的全局变量，可以用来传输用户数据（参见 GUIDATA）
```

在上面这段代码后添加以下代码：

```
a = get(handles.edit1,'String');
b = get(handles.edit2,'String');
% a 和 b 是字符串类型，需要转换为数字类型的变量才能相加
total = str2num(a) + str2num(b);
c = num2str(total);
% 需要将运算结果转换回字符串类型然后加以显示
set(handles.text8,'String',c);
guidata(hObject, handles);
```

下面对上面这段程序做一下分析。

```
a = get(handles.edit1,'String');
b = get(handles.edit2,'String');
```

上面这行代码把用户输入的数据存入到变量 *a*、*b* 中。

a 和 *b* 是字符型变量，需要转换为数字类型的变量才能相加。

```
total = str2num(a) + str2num(b);
```

这段代码实现两数相加。

```
c = num2str(total)。
set(handles.text8,'String',c);guidata(hObject, handles);
```

分别用来更新计算结果文本框和图形对象句柄，一般 Callback 回调函数都以 guidata(hObject, handles)结束已更新数据。

程序运行结果如图 17-9 所示。



图 17-9　运行结果

17.3　对话框对象

在 GUI 程序设计中，对话框是最重要的显示信息和取得用户数据的用户界面对象。对话框一般包含一个或多个按钮以供用户输入或者弹出显示的信息，由用户决定要采取的措施。

在绝大多部分的程序设计软件中，如 Visual Basic、Delphi、Visual C++，都可以方便地进行对话框设计。MATLAB 也提供了一些进行对话框设计的有用函数。使用对话框，可以使图形用户界面更加友好，易于为用户理解。

总的来说，MATLAB R2018a 中的对话框分为两大类，第一类是公共对话框，第二类是一般对话框。

下面对 MATLAB R2018a 中的对话框对象进行介绍。

17.3.1 公共对话框

常用的公共对话框函数如表 17-4 所示。

表 17-4 常用的公共对话框函数

对话框函数	说　明	对话框函数	说　明
uigetfile	标准的打开对话框	pagesetupdlg	页面设置对话框
uiputfile	保存文件对话框	printpreview	打印预览对话框
uisetfont	生成字体和字体属性选择对话框	printdlg	打印对话框
uisetcolor	颜色设置对话框		

下面分别对它们进行介绍。

1. 文件打开对话框

文件打开对话框用于打开某个文件。在 Windows 系统中，几乎所有的应用软件都提供了文件打开对话框。在 MATLAB 中，调用文件打开对话框的函数为 uigetfile。该函数的调用格式如表 17-5 所示。

表 17-5 文件打开对话框函数的调用格式

格　式	说　明
uigetfile	在显示的文件打开对话框中，列出当前目录下 MATLAB 能识别的所有文件
uigetfile('FilterSpec')	在显示的文件打开对话框中，列出当前目录下由参数 FilterSpec 指定的类型文件。参数 FilterSpec 是一个文件类型过滤字符串，用于指定要显示的文件类型。例如，'*.m'显示当前目录下 MATLAB 中的所有 M 文件
uigetfile('FilterSpec', 'DialogTitle')	与第二种调用格式基本相同，只是该调用格式设定了文件打开对话框的标题名，默认标题为字符串 Select file to open
uigetfile('FilterSpec', 'DialogTitle',x,y)	与第三种调用格式基本相同，只是该调用格式还指定了对话框显示时的位置。显示位置由参数 *x* 与 *y* 决定。*x*、*y* 是从屏幕左上角算起的水平与垂直距离。距离单位是像素
[fname,pname]=uigetfile(…)	返回打开文件的文件名与路径。其中，输入参数 fname 存放的是打开的文件名，pname 包含文件的路径

2. 文件保存对话框

文件保存对话框用于保存某个文件。在 MATLAB 中，调用文件保存对话框的函数为 uiputfile。该函数的调用格式如表 17-6 所示。

表 17-6　文件保存对话框函数的调用格式

格　　式	说　　明
uiputfile	显示用于保存文件的对话框，列出当前目录下 MATLAB 能识别的所有文件
uiputfile ('InitFile')	在显示的文件保存对话框中，列出当前目录下由参数 InitFile 指定的类型文件。参数 InitFile 是一个文件类型过滤字符串，用于指定要保存的文件类型。例如，'*.m' 显示当前目录下 MATLAB 中的所有 M 文件
uiputfile ('InitFile', 'DialogTitle')	与第二种调用格式基本相同，只是该调用格式设定了文件保存对话框的标题名，默认标题为字符串 Select file to write
uiputfile ('InitFile', 'DialogTitle',x,y)	与第三种调用格式基本相同，只是该调用格式还指定了对话框显示时的位置。显示位置由参数 *x* 与 *y* 决定。*x*、*y* 是从屏幕左上角算起的水平与垂直距离。距离单位是像素
[fname,pname]=uiputfile(…)	返回保存的文件的文件名与路径。其中，输入参数 fname 存放的是保存的文件名，pname 包含文件的路径

3．颜色设置对话框

颜色设置对话框可用于交互式设置某个图形对象的前景色或背景色。在绝大部分的程序设计软件中，都提供了这个公共对话框。在 MATLAB 中，调用颜色设置对话框的函数为 uisetcolor。该函数的调用格式如表 17-7 所示。

表 17-7　颜色设置对话框函数的调用格式

格　　式	说　　明
c=uisetcolor(h_or_c, 'DialogTitle')	输入参数中的 h_or_c 可以是一个图形对象的句柄，也可以是一个三色 RGB 向量。若是图形对象的句柄，那么该图形对象必须有一个颜色属性；若是三色 RGB 向量，那么输入的必须是一个有效的 RGB 向量，此时输入的颜色初始化颜色对话框。对于输入的参数是图形对象句柄，颜色对话框被初始化为黑色。输入参数中的 DialogTitle 字符串用于标明颜色对话框的标题名。输出参数 c 返回用户选择的 RGB 向量值

4．字体设置对话框

字体设置对话框可用于交互式修改文本字符串、坐标轴或控件对象的字体属性。可以修改的字体属性包括 FontName、FontUnits、FontSize、FontWeight、FontAngle 等。在 MATLAB 中，调用字体设置对话框的函数为 uisetfont。该函数的调用格式如表 17-8 所示。

表 17-8　字体设置对话框函数的调用格式

格　　式	说　　明
uisetfont	显示用于进行字体设置的对话框，对话框中列出了字体、字形、字体大小等字段。返回的是选择的字体的属性值
uisetfont('DialogTitle')	与第一种调用格式基本相同，只是该调用格式设定了字体设置对话框的标题名，默认标题为字符串 Font
uisetfont(h)	输入参数 h 是一个对象句柄。该调用格式用对象句柄中的字体属性值初始化字体设置对话框中的属性值，用户可以利用字体设置对话框重新设置对象的字体属性值。返回重新设置后的字体的属性值
uisetfont(h, 'DialogTitle')	与第三种即 uisetfont(h)的调用格式基本相同，只是该调用格式还设定了字体设置对话框的标题名，默认标题为字符串 Font
uisetfont(S)	输入参数 S 是一个字体属性结构，是一个或多个下列属性的合法值：FontName、FontUnits、

（续表）

格　式	说　明
	FontSize、FontWeight、FontAngle，否则输入值会被忽略。该调用格式用字体属性结构 S 中的成员值来初始化字体设置对话框中的属性值。用户可以利用字体设置对话框重新设置对象的字体属性值。返回重新设置后的字体的属性值
uisetfont(S,'DialogTitle')	与第五种即 uisetfont(S)的调用格式基本相同，只是该调用格式还设定了字体设置对话框的标题名，默认标题为字符串 Font
S= uisetfont(…)	返回字体属性（FontName、FontUnits、FontSize、FontWeight、FontAngle）的属性值，被保存在结构 S 中

5．打印页面设置对话框

打印页面设置对话框可用于对打印输出时的页面进行设置。在许多应用软件中，都提供了进行打印页面设置的对话框。在 MATLAB 中，调用打印页面设置对话框的函数为 pagesetupdlg。该函数的调用格式如表 17-9 所示。

表 17-9　打印页面设置对话框函数的调用格式

格　式	说　明
dlg=pagesetupdlg(fig)	用默认的页面设置属性为图形窗口创建一个打印页面设置对话框。输入参数 fig 必须是单个图形窗口的句柄，而不是一个图形窗口向量。若省略参数 fig，那么默认的图形窗口对象是当前窗口对象。输出参数返回已设置了的打印页面属性值

在 MTALAB R2018a 中，pagesetupdlg 函数被打印预览对话框函数 printprview 合并，具体见表 17-10 所示。

6．打印预览对话框

打印预览对话框用于对打印输出的页面进行预览。在许多应用软件中，都提供了进行打印预览的对话框。在 MATLAB 中，调用打印预览对话框的函数为 printpreview。该函数的调用格式如表 17-10 所示。

表 17-10　打印预览对话框函数的调用格式

格　式	说　明
printprview	显示当前图形窗口对象的打印预览对话框
printprview(fig)	显示指定的图形窗口对象 fig 的打印预览对话框

如图 17-10 所示为对当前图形窗口对象内的图形进行预览。

在打印预览对话框的上面有相应的分组按钮控件，用户可以进行相应的编辑控制操作。

7．打印对话框

打印对话框是专门进行打印的对话框，这是任何图形用户界面的应用软件都会提供的对话框。在 MATLAB 中，调用打印对话框的函数为 printdlg。该函数的调用格式如表 17-11 所示。

表 17-11　打印对话框函数的调用格式

格　　式	说　　明
printdlg	显示出标准的 Windows 打印对话框（在 Windows 系统中）。它打印当前图形窗口内的图形对象
printdlg(fig)	显示出标准的 Windows 打印对话框（在 Windows 系统中），但它打印由输入参数 fig 指定的图形窗口内的对象。输入参数 fig 是将要打印的图形窗口的句柄
printdlg('-crossplatform',fig)	显示标准的 crossplatform 模式的 MATLAB 打印对话框。输入参数 fig 是将要打印的图形窗口的句柄
printdlg('-setup',fig)	显示打印对话框的开始模式。在这里有可以不进行打印而设置的默认选项

打印对话框如图 17-11 所示。

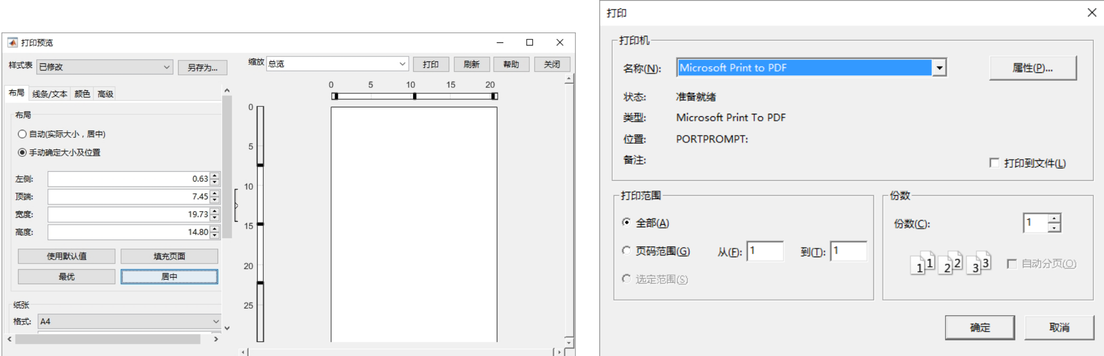


图 17-10　打印预览对话框　　图 17-11　打印对话框

17.3.2　一般对话框

除了提供大量标准的公共对话框，MATLAB R2018a 还提供了大量的一般对话框函数，如表 17-12 所示。

表 17-12　一般对话框函数

对话框函数	说　　明	对话框函数	说　　明
helpdlg	帮助对话框	listdlg	列表选择对话框
errordlg	错误消息对话框	axlimdlg	生成坐标轴范围设置对话框
msgbox	信息提示对话框	dialog	创建对话框或图形用户对象类型的图形窗口
questdlg	询问对话框	menu	菜单类型的选择对话框
warndlg	警告消息显示对话框	waitbar	显示等待进度条
inputdlg	变量输入对话框		

下面分别对它们进行介绍。

1．帮助对话框

在操作应用软件时，当用户不知道该如何操作时，帮助信息将会帮助用户进行正确操作，显示帮助信息的对话框就是帮助对话框。MATLAB 提供的创建帮助对话框的函数是 helpdlg。该函数的调用格式如表 17-13 所示。

表 17-13　帮助对话框函数的调用格式

格　式	说　明
helpdlg	创建一个默认的帮助对话框。默认的对话框名为“Help Dialog”，在对话框内包含一个名为“This is the default help string”的字符串
helpdlg('helpstring')	创建一个帮助对话框。该对话框仍然叫作“Help Dialog”，但对话框内显示的帮助信息由输入参数 helpstring 决定
helpdlg('helpstring', 'dlgname')	创建一个帮助对话框。该对话框的名称由输入参数 dlgname 决定，对话框内显示的帮助信息由输入参数 helpstring 决定
h= helpdlg(…)	返回创建的帮助对话框的句柄，句柄存放在输入参数 h 中，输入参数与前面的调用格式的输入参数相同

MATLAB 会自动设置帮助对话框的宽度，使其能够显示出 helpstring 字符串的全部帮助信息。例如：

```
helpdlg('从当前图形中选取 9 个点！','选择取');
```

相应得到的帮助对话框如图 17-12 所示。



图 17-12　帮助对话框

2. 错误消息对话框

在开发的应用软件中，当用户进行了错误的操作后，应该显示错误消息对话框，使用户知道错误的原因，以便采取正确的操作。此时，就要用到错误消息对话框。MATLAB 提供的创建错误消息对话框的函数是 errordlg，该函数的调用格式如表 17-14 所示。

表 17-14　错误消息对话框函数的调用格式

格　式	说　明
errordlg	创建一个默认的错误消息对话框。默认的对话框名为“Error Dialog”，在对话框内包含一个名为“This is the default error string”的字符串
errordlg ('errorstring')	创建一个错误消息对话框。该对话框仍然叫作“Error Dialog”，但对话框内显示的错误消息由输入参数 errorstring 决定
errordlg('errorstring', 'dlgname')	创建一个错误消息对话框。该对话框的名称由输入参数 dlgname 决定，对话框内显示的帮助信息由输入参数 errorstring 决定
errordlg('errorstring', 'dlgname', 'on')	创建一个错误消息对话框。当要创建的对话框已经存在时，输入参数 on 把已经存在的对话框显示在屏幕的最前端，而不再创建同名的新对话框
h=errordlg(…)	返回创建的错误消息对话框的句柄，句柄存放在输入参数 h 中。输入参数与前面的调用格式的输入参数相同

MATLAB 会自动设置错误消息对话框的宽度，使其能够显示出 errorstring 字符串的全部错误信息。显示的错误消息对话框的外观依赖于不同的操作系统。

例如，弹出“未找到文件”对话框。

```
errordlg('未找到文件','文件错误');
```

相应得到的错误消息对话框如图 17-13 所示。

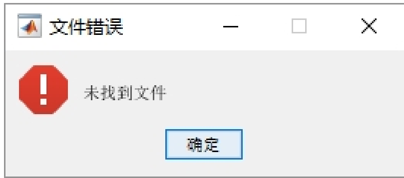


图 17-13　错误消息对话框

3．信息提示对话框

在面临多种选择或者应该显示某种提示情况时，一般就会显示信息提示，此时就要借助于信息提示对话框。MATLAB 提供的创建信息提示对话框的函数是 msgbox。该函数的调用格式如表 17-15 所示。

表 17-15　信息提示对话框函数的调用格式

格　　式	说　　明
msgbox(message)	创建一个信息提示对话框。创建的对话框会自动设置对话框的宽度，使其能够显示出全部提示信息。输入参数 message 存储的是要显示的提示信息。该参数取值可以是一个字符串向量或字符串矩阵
msgbox(message,title)	创建的信息提示对话框与第一种格式即 msgbox(message)创建的对话框基本相同，只是该对话框有一个标题名，标题名由输入参数 title 决定。参数 title 是一个字符串
msgbox(message,title, 'icon')	创建的信息提示对话框除了包含提示信息与标题名，对话框上还有一些图标。图标由参数 icon 决定，icon 可选的值有 none、error、help、warn、custom，默认值是 none
msgbox(message,title, 'custom',iconData,iconCmap)	创建的信息提示对话框中的图标是用户自定义的图标。定义图标的图像数据存放在参数 iconData 中，定义图像的颜色数据存放在参数 iconCmap 中
Msgbox=(…, 'createMode')	参数 createMode 用于决定创建的信息提示对话框是模式对话框还是无模式对话框。参数 createMode 的可选值有 modal、non-modal 和 replace。其中，replace 值用标题名相同的对话框代替另外一个已经打开的对话框
H= msgbox(…)	返回创建的信息提示对话框的句柄，句柄存放在输入参数 h 中。输入参数与前面的调用格式的输入参数相同

例如，创建一个名为“注意”的信息提示对话框，显示的提示信息是字符串“这是第一个信息提示对话框！”。对话框上有一个警告的图标。它是无模式对话框。

相应命令为：

```
msgbox('这是第一个信息提示对话框!','注意','warn','non-modal');
```

结果如图 17-14 所示。



图 17-14　信息提示对话框

4．询问对话框

当对问题的解决可能存在多种选择时，就会显示一个询问对话框，由用户决定应该采取的步骤。例如，保存文件的文件名与当前目录中存在的某个文件名相同时，就会显示询问对话框。MATLAB 提供的创建询问对话框的函数是 questdlg。该函数的调用格式如表 17-16 所示。

表 17-16　询问对话框函数的调用格式

格　式	说　明
button=questdlg('qstring')	创建一个问题显示的模式对话框。该对话框中有 3 个命令按钮，分别为 Yes、No、Cancel。显示的问题由输入参数 qstring（字符串类型）决定。输出参数 button 返回的是用户按下的命令按钮的名字
button=questdlg('qstring', 'title')	创建的询问对话框的标题由参数 title 决定。该标题显示在对话框的标题栏中
button=questdlg('qstring', 'title', 'default')	创建的询问对话框，当用户按下 Enter 键时，返回参数 button 中的值是参数 default 设置的值。default 必须是 Yes、No、Cancel 中的一个
button=questdlg('qstring', 'title', 'str1', 'str2', 'default')	创建的询问对话框有两个命令按钮，按钮上显示的字符由参数 str1 与 str2 决定。default 设置当用户按下 Enter 键时返回的参数值。default 必须是 str1、str2 中的一个
button=questdlg('qstring',title','str1','str2','str3','default')	创建的询问对话框中有 3 个命令按钮，按钮上显示的字符由参数 str1、str2 与 str3 决定。default 设置当用户按下 Enter 键时返回的参数值。default 必须是 str1、str2 或 str3 中的一个

MATLAB 会自动设置询问对话框的宽度，使其能够显示出 qstring 字符串的全部信息。

例如，命令行输入：

```
button=questdlg('想继续吗?!','继续操作','是','否','帮助','否');
```

结果如图 17-15 所示。



图 17-15　询问对话框

5．警告消息显示对话框

在操作应用软件时，当用户进行了不恰当的操作后，应该显示警告消息显示对话框，使用户知道该操作可能导致错误，以便采取正确的操作。此时，就要用到警告消息显示对话框。MATLAB 提供的创建警告消息显示对话框的函数是 warndlg。该函数的调用格式如表 17-17 所示。

表 17-17　警告消息显示对话框函数的调用格式

格　式	说　明
warndlg	创建一个默认的警告消息显示对话框。默认的对话框名为“warning Dialog”，在对话框内包含一个名为“This is the default warning string”的字符串
warndlg('warnstring')	创建一个警告消息显示对话框。该对话框仍然叫作“warning Dialog”，警告信息由输入参数 warnstring 决定

（续表）

格　　式	说　　明
warndlg('warnstring', 'dlgname')	创建一个警告消息显示对话框。该对话框的名称由输入参数 dlgname 决定，对话框内显示的警告信息由输入参数 warnstring 决定
warndlg('warnstring', 'dlgname', 'on')	创建一个警告消息显示对话框。当要创建的对话框已经存在时，输入参数 on 把已经存在的对话框显示在屏幕的最前端，而不再创建同名的新对话框
H= warndlg(…)	返回创建的警告消息显示对话框的句柄，句柄存放在输入参数 h 中。输入参数与前面的调用格式的输入参数相同

MATLAB 会自动设置警告消息显示对话框的宽度，使其能够显示出 warnstring 字符串的全部警告信息。显示的警告消息对话框的外观依赖于不同的操作系统。

例如，创建一个名为“!! 警告!!”的对话框，其中显示的警告消息是字符串“单击 OK 按钮将会造成所有文件损坏”。

在命令行输入：

```
warndlg('单击 OK 按钮将会造成所有文件损坏','!!警告!!')
```

结果如图 17-16 所示。

6．变量输入对话框

在许多应用软件中，当需要用户输入变量时，就会显示一个输入对话框。MATLAB 提供的创建变量输入对话框的函数是 inputdlg。该函数的调用格式如表 17-18 所示。

表 17-18　变量输入对话框函数的调用格式

格　　式	说　　明
answer=inputdlg(prompt)	创建一个模式变量输入对话框。输入参数是提示输入信息的字符串。返回值 answer 存储用户输入的变量值
answer=inputdlg(prompt,title)	创建的模式变量输入对话框的标题名由参数 title 决定。该参数是一个字符串
answer=inputdlg(prompt,title,lineNo)	创建的模式变量输入对话框中用于输入变量的可编辑文本框的行数由 lineNo 决定。该参数可以是标量、列向量或矩阵。lineNo 的默认值是 1。有多个可编辑文本框时，用户输入的值都存储在参数 answer 中，只是此时要求 answer 是一个向量值
answer=inputdlg(prompt,title,lineNo,defAns)	创建的变量输入对话框的每个可编辑文本框中的默认值由参数 defAns 决定。defAns 的值就显示在每个可编辑文本中。参数 defAns 是一个字符向量，其元素的个数必须与参数 prompt 中元素的个数相等
answer=inputdlg(prompt,title,lineNo,defAns,Resize)	输入参数用于决定创建的变量输入对话框的大小能否被调整。若取值是字符串 on，那么创建的对话框的大小可以被调整；若取值是字符串 off，那么创建的对话框的大小不能被调整

例 17-4，创建一个变量输入对话框，可以输入一个整数与一个颜色名。其中，第一个可编辑文本框的行数为 2 行，其默认值为 20；第二个可编辑文本框的行数是 4 行，其默认值是 HSV。

在命令行窗口中输入：

```
prompt={'输入矩阵大小:','输入色彩模型名:'};
title='峰函数输入';
```

```
lines=[2 4]';
def={'20','HSV'};
answer=inputdlg(prompt,title,lines,def);
```

结果如图 17-17 所示。



图 17-16　警告消息显示对话框



图 17-17　变量输入对话框

7. 列表选择对话框

当存在多个选项时，最好提供给用户一个列表框，把所有可能的选项都列出来，使用户能从中选择一个需要的值。在这种情况下就要用到列表选择对话框。MATLAB 提供的创建列表选择对话框的函数是 listdlg。该函数的调用格式如表 17-19 所示。

表 17-19　列表选择对话框函数的调用格式

格　　式	说　　明
[Selection,ok]= listdlg('ListString',S, …)	创建一个列表选择模式对话框。从列表框中可以选择一个或多个列表项。输出变量是一个向量，存储的是选择的列表项的索引号。在单模式选择状态下，它的长度是 1。当输出变量 OK 是 0 时，变量 Selection 是空向量[]。若用户单击列表选择对话框中的 OK 按钮，则输出变量 OK 值是 1；若单击该对话框中的 Cancel 按钮或关闭列表选择对话框，则输出变量 OK 值是 0。双击选择的列表项或选中列表项，然后单击对话框中的 OK 按钮，其效果一样。在多模式选择状态下，单击对话框中的 Select all 按钮，可以选择所有的列表项

列表选择对话框中的输入参数可以选取表 17-20 中所列的值。

表 17-20　列表选择对话框中的输入参数

参　　数	参数功能简介
'ListString'	该参数用于设置列表框中的列表项，是一个字符向量
'SelectionMode'	该参数用于设置是单模式还是多模式选择，默认是多模式选择。参数取值是一个字符串，可以取 single 或 multiple
'LiseSize'	该参数用于设置列表框的大小，是一个两元素向量[width height]，默认值是[160 300]，单位是像素
'InitialValue'	该参数用于说明初始选择的列表项的索引，默认是 1，即第一项。参数的取值是一个向量
'Name'	该参数用于设置列表选择对话框的标题名。默认是空字符串
'PromptString'	该参数是一个字符矩阵或字符向量，用于标明在列表框上面的说明性文本。默认是空
'OKString'	该参数用于设置 OK 按钮上的文本。默认是字符串 OK
'CancelString'	该参数用于设置 Cancel 按钮上的文本。默认是字符串 Cancel
'uh'	该参数用于设置控件按钮的高度。默认是 18，单位是像素
'fus'	该参数用于设置框架与控件对象之间的距离。默认是 8，单位是像素
'ffs'	该参数用于设置框架与图形窗口对象之间的距离。默认是 8，单位是像素

如在命令行窗口中输入：

```
d=dir;
str={d.name};
[s,v]=listdlg('Promptstring','选择文件:','selectionmode','single',
   'listsize', [300 400],'liststring',str);
```

结果如图 17-18 所示。

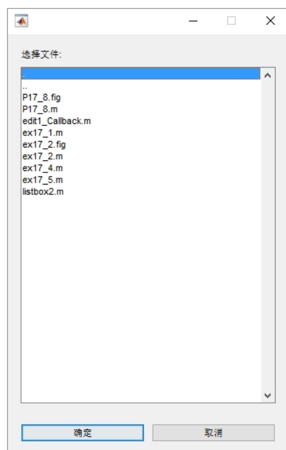


图 17-18　列表选择对话框

除了前面介绍的对话框函数，MATLAB R2018a 还提供了一些其他的对话框函数，例如用于创建对话框或图形用户对象类型的图形窗口的 dialog 函数、菜单类型的选择对话框 menu 函数及显示等待进度条的 waitbar 函数等。关于它们的创建方法，可以参考 MATLAB 帮助文件。

17.4　界面菜单

在绝大多数的图形用户界面（GUI）下，都包含菜单。通过选择各级菜单，可以执行相应的命令，实现相应的功能。一般地，从菜单的标题或名字可以大概了解该菜单的功能。

同样，使用 MATLAB R2018a 来创建图形用户界面时，也常常需要产生界面菜单；在编辑其他编辑对象时，需要产生弹出菜单，即上下文菜单。

这两种菜单在 MATLAB R2018a 中可以通过 uimenu 或 uicontextmenu 来产生。下面对 MATLAB 中的菜单进行介绍。

17.4.1　菜单建立

在 MATLAB R2018a 中，可以通过命令行和 GUI 设计工具的菜单编辑器 Menu Editor 两种方式来建立界面菜单。

1. 命令行方式

在命令行方式下，可以通过函数 uimenu 来建立下拉式菜单对象。uimenu 函数常见的命令格式如表 17-21 所示。

表 17-21　uimenu 函数常见的命令格式

格　　式	说　　明
uimenu('PropertyName', PropertyValue,…)	使用指定的属性值在当前图形窗口产生菜单，可以产生的属性值后面将做介绍
uimenu(parent,'PropertyName', PrpertyValue,…)	在父菜单中产生子菜单，或在由父窗口指定的上下文菜单中产生菜单项。如果父句柄指向另一个 uimenu 对象或 uicontextmenu 对象的图形，那么将会产生新的菜单栏
handle =uimenu('PropertyName', PropertyValue,…)	可以返回菜单句柄，其余的设置同上
handle =uimenu(parent,'PropertyName',PropertyVaLue,…)	可以返回菜单句柄

在命令行方式下，可以通过函数 uicontextmenu 来创建内容式菜单对象。uicontextmenu 函数常见的命令格式如表 17-22 所示。

表 17-22　uicontextmenu 函数常见的命令格式

格　　式	说　　明
Hm_contextmenu=uicontextmenu ('PropertyName', PropertyValue,…)	Hm_contextmenu 是创建的菜单项的句柄值；PropertyName 是菜单的某个属性的属性名；PropertyValue 是与菜单属性名相对应的属性值

利用 uicontextmenu 函数生成内容式菜单后，再通过函数 uimenu 可在以往创建的内容式菜单中添加子菜单。然后可以通过 set 把创建的内容式菜单与某个对象相联系，通过设置对象的 uicontextmenu 属性，使内容式菜单依附于该对象。

2. GUI 设计工具菜单编辑器

在命令行方式下创建菜单时，需要记住用于创建的函数，而且要记住大量菜单属性的属性名，这对于创建菜单非常不方便。

在 MATLAB R2018a 中利用 GUI 设计工具中的菜单编辑器（Menu Editor），可以很方便地创建下拉式菜单与内容式菜单。

17.4.2　菜单属性

利用函数 uimenu 或 uicontextmenu 建立菜单时，可以定义菜单属性的属性值。利用函数 set，可以设置、改变属性的属性值；利用函数 get，可以获得菜单属性的属性值，也可以通过 Property Inspector GUI 设计工具来设置、改变菜单的属性值。

1．uimenu 菜单的属性

在菜单中，uimenu 函数的属性名称和属性值的常用设置如表 17-23 所示。

表 17-23　uimenu 函数的属性名称和属性值

属性名称（PropertyName）	属性值（PropertyValue）	说　　明
外观及风格控制属性		
Checked	Value：on，off Default：off	菜单项前是否添加核选标记“√”
Label	Value：String	设置菜单的标题名称

（续表）

属性名称（PropertyName）	属性值（PropertyValue）	说　　明
Separator	Value：on，off Default：off	分隔条
Foregroundcolor	Value：ColorSpec Default：[0 0 0]	文本颜色
Visible	Value：on，off Default：on	控制 uimenu 菜单的可见状态
	基本信息属性	
Accelerator	Value：character	键盘加速键
Children	Value：Vectorofhandles	子菜单句柄
Enable	Value：cancel，queue Default：queue	禁止或使用 uimenu
Parent	Value：handle	菜单对象的父对象
Tag	Value：String	用户指定的对象标识符
Type	Value：String(read-only) Default：uimenu	图形对象的类
UserData	Value：matrix	用户指定的数据
	位置信息属性	
Position	Value：scalar Default：[1]	相对的 uimenu 的位置
	回调函数执行控制属性	
BusyAction	Value：cancel，queue Default：queue	回调函数中断
Callback	Value：string	控制动作
CreateFcn	Value：string	在对象生成过程中执行回调
DeleteFcn	Value：string	在对象删除过程中执行回调
Interruptible	Value：on，off Default：on	回调函数的中断方式
	控制操作属性	
Handle Visibility	Value：on，callback，off Default：on	在命令行或 GUI 中是否可见

2. uicontextmenu 菜单的属性

参见 uimenu 菜单的属性。

在命令行方式或者 GUI 设计方式下，用户可以方便地对 uimenu 菜单与 uicontextmenu 菜单的属性进行设置、修改。

17.5　编写 M 文件

通过前面几节的介绍，读者可以通过不断地开发来熟悉和掌握使用 MATLAB R2018a 的图形用户界面函数来创建界面图形的方法。

然而，如果只在 MATLAB 命令行窗口中输入命令来编写 GUI 界面，那么编程的效率实在是太低了。本节将结合实例介绍如何编写 GUI 的 M 文件。

M 文件中所有的代码，包括命令、变量等，都在 MATLAB 命令行窗口中执行。因此可以随时使用所有在脚本式 M 文件中编写的函数、变量和对象句柄等，并将它们传递给其他函数。

例 17-5，编写一个脚本式 M 文件。在当前图形窗口中创建一个列表框。列表框中有“第一个”、“第二个”和“第三个”3 个字符串。当用户选择任何一个列表项后，就调用编写的脚本式 M 文件，并显示相应的信息。

首先在 MATLAB R2018a 的命令行窗口中输入命令：

```
 str(1)={'第一个'};
    str(2)={'第二个'};
    str(3)={'第三个'};
    Hm_list=uicontrol('style','listbox','position',[200 250 80 100],
                   'string',str, 'callback','listbox1_callback');
```

可生成如图 17-19 所示的对话框。

接着编写脚本式 M 文件，该文件的文件名为 listbox1_callback.m，供创建的列表框的 Callback 属性调用。

```
index_selected=get(Hm_list,'Value');
list=get(Hm_list,'String');
listnum=list{index_selected}
switch listnum
  case '第一个'
     msgbox('这是第一个对话框！','消息','non-modal');
  case '第二个'
     msgbox('这是第二个对话框！','消息','non-modal');
  case '第三个'
     msgbox('这是第三个对话框！','消息','non-modal');
  otherwise
     msgbox('未选中选项，请选择！','消息','non-modal');
end
```

最终运行结果如图 17-20 所示。

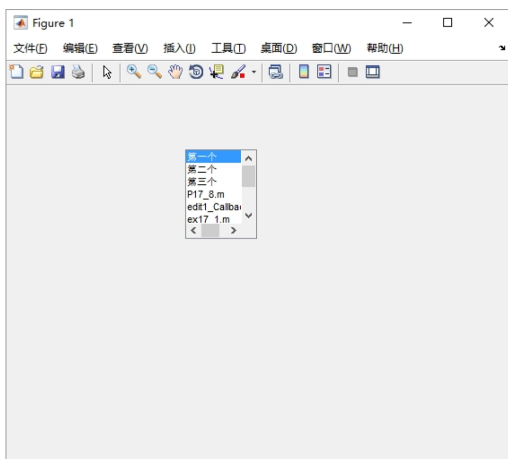


图 17-19　生成对话框

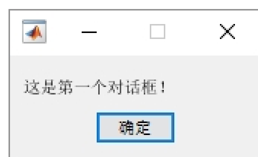


图 17-20　运行结果

17.6　图形界面创建工具 GUIDE

前面章节中对用户图形界面的创建命令和方法进行了介绍，只涉及其中的一部分内容。通过这些内容的学习，读者可以很容易掌握使用命令函数方式来创建用户界面的方法。关于这部分内容，读者可以通过阅读相关的帮助文件获得更为详细的帮助内容。

此外，MATLAB R2018a 还提供了功能强大的 GUIDE 模板，用于创建各种用户图形对象。同时，通过该 GUIDE 模板的使用，还可以自动生成相应的 M 文件框架，简化了用户编写图形界面程序的工作。如果读者需要更改相关的内容，则可以直接打开 M 文件进行修改或编写自己的代码。

因此，可以说，图形界面创建工具 GUIDE 大大增强了图形界面创建的能力。

17.6.1　利用创建工具 GUIDE 进行 GUI 设计

本节将通过实际的操作说明利用创建工具 GUIDE 进行 GUI 设计的步骤和方法。

例 17-6，GUIDE 图形界面设计实例：创建图形用户界面用于对不同的数据进行可视化显示。

1．GUIDE 的启动

用户可以在 MATLAB R2018a 的命令行中输入 guide，来启动图形界面创建工具 GUIDE，如图 17-21 所示。

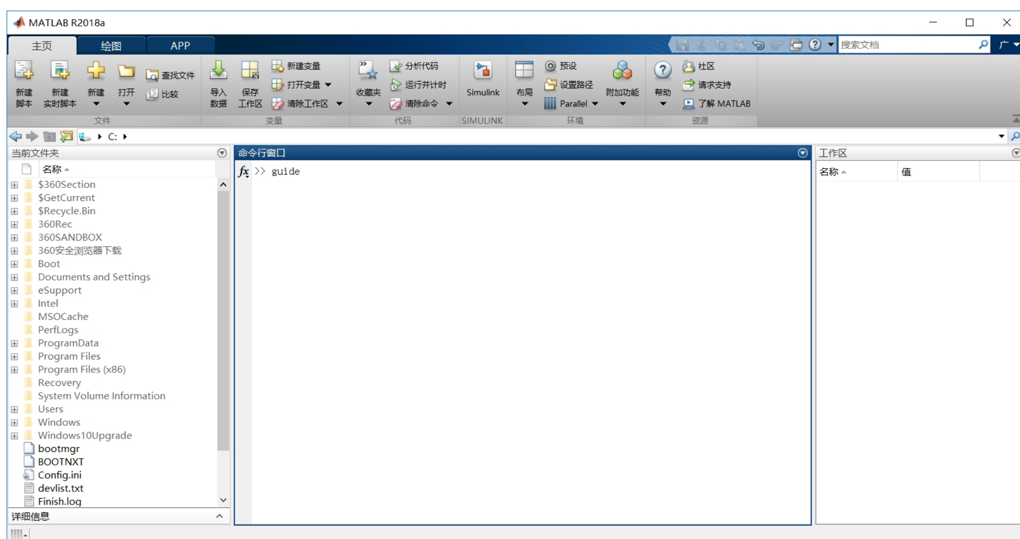


图 17-21　启动 GUIDE

启动后，系统弹出如图 17-22 所示的 GUIDE 快速入门对话框。该对话框中主要包括两个选项卡，其中一个为创建新的用户图形界面（新建 GUI）。在该选项卡中包含了 4 种类型的 GUI 模板：默认的空白 GUI 模板（Blank GUI）、带有 Uicontrol 控件的 GUI 模板、带有坐标轴和菜单的 GUI 模板及模态对话框模板。

可以选择的另外一个选项卡为打开已有的 GUI 对象，或者单击该选项卡中的浏览按钮来浏览选择已经存在的 GUI 对象。

此处，选择使用空白的 GUI 模板来创建图形用户界面。选择该对象后，将会出现一个非常短暂的初始化过程，初始化完毕后将显示如图 17-23 所示的空白图形用户界面对象的编辑界面。

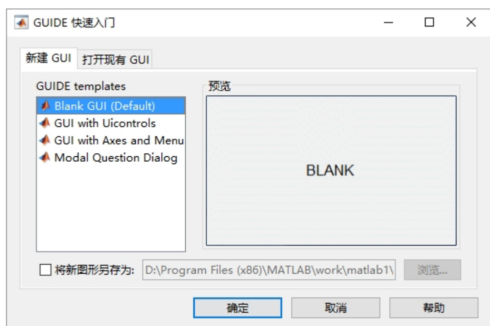


图 17-22　GUIDE 快速入门对话框

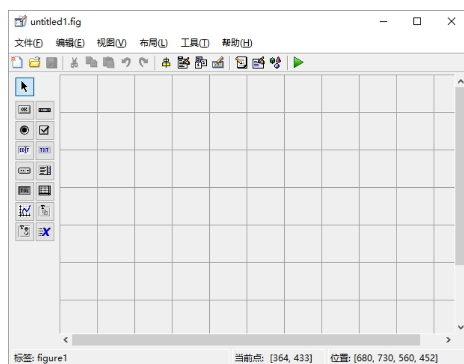


图 17-23　空白图形用户界面对象的编辑界面

如果用户需要对该编辑界面的显示属性进行修改或重新设置，则可以在该编辑界面选择文件→预设选项命令进行设置。

预设项对话框如图 17-24 所示，可以设置的属性包括在组件选项板中显示名称、在窗口标题中显示文件扩展名、在窗口标题中显示文件路径及为新生成的回调函数添加注释等。如果用户需要设置其中的一些选项，那么将对应的复选框选中即可。

此处，对比图 17-23，如果选中在组件选项板中显示名称复选框（即图 17-24 中的第一个复选框），那么将会在选择的组件栏中显示组件的名称，如图 17-25 所示。显示组件名称可以更好地帮助用户熟悉组件对象的图标，更方便地完成组件的选择和使用。

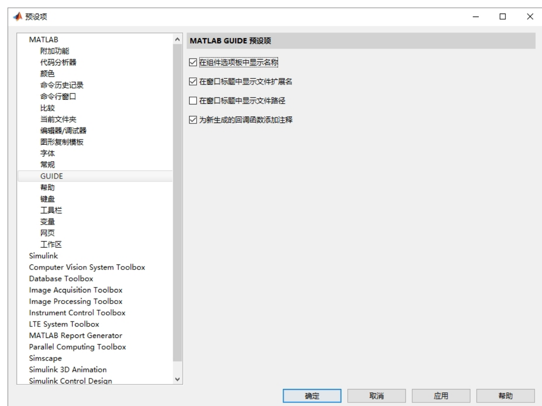


图 17-24　设置 GUIDE 编辑界面的预设项对话框

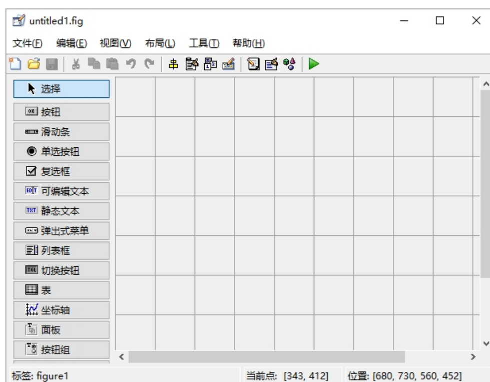


图 17-25　改变预设显示组件名称

2. 创建图形用户界面对象

接下来可以在打开的图形用户界面上添加控件来完成这个图形界面的布局和控制。此处将通过对不同函数的显示和处理方式来说明如何使用 GUIDE 来创建新的图形用户界面（GUI）。

在前面介绍的内容的基础上，用户可以在图形界面编辑对话框中，单击该界面的右下角，用鼠标拖动以改变界面的大小，如图 17-26 所示。

如果需要精确控制界面的大小和范围，则可以单击属性检测器按钮，系统将弹出检查器对话框，如图 17-27 所示。在该对话框中，选择 Units 属性，在弹出的列表中选择 Characters 选项；然后在 Position 属性中设置图形用户界面的宽和高。

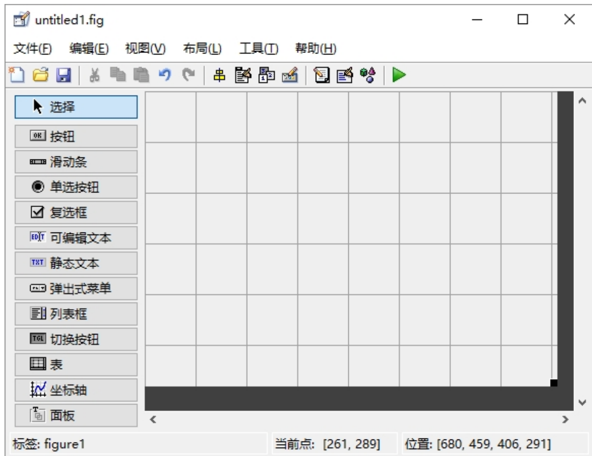


图 17-26　鼠标拖动改变图形界面的大小

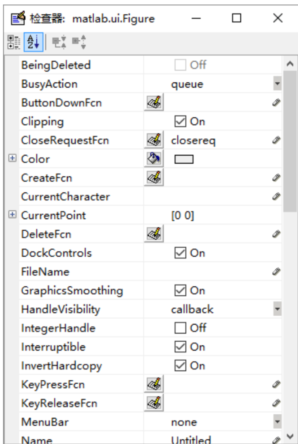


图 17-27　GUIDE 图形编辑界面的属性检查器

接下来，向图形编辑界面上添加 3 个按钮对象。添加时，在 GUIDE 编辑对话框的左侧单击组件后，在编辑界面上单击即可将该对象放在边界界面上。添加其他对象的方式与此相同。添加完毕后的效果如图 17-28 所示。

在图形编辑界面中添加面板对象，并将前面添加的 3 个对象一起拖到面板中。设置完毕后的效果如图 17-29 所示。

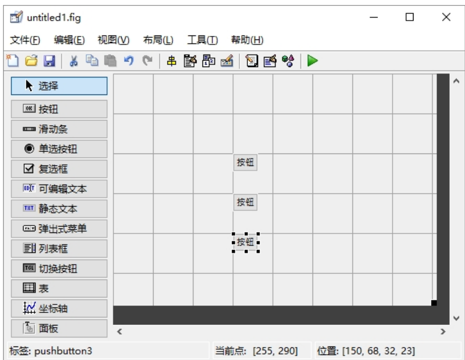


图 17-28　添加按钮控件

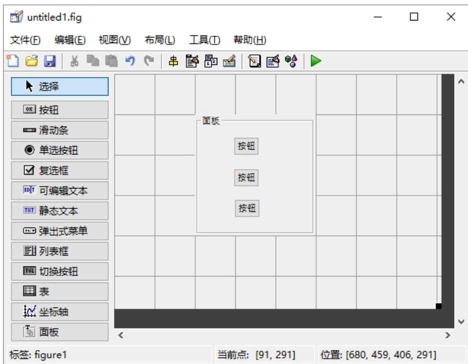


图 17-29　将按钮控件添加到面板中

如果按钮对象没有对齐，此时，可以在工具栏中单击对齐图标对齐所选择的对象，系统会弹出“对齐对象”对话框，如图 17-30 所示。在该对话框中可以设置水平方向和垂直方向的对齐方式及对象的分布间隔。此时，可以按照对话框中的说明来选择和设置必要的对齐方式及分布间隔。

除上面的对齐调整外，用户还可以调整对象的位置。如调整 3 个按钮的位置时，可以在选中 3 个按钮后，用方向键进行微调。其他对象的调整方法与此相同。

当对上述对象进行调整之后，可以将面板及面板上的 3 个按钮一同移动到图形用户界面的适当位置。

以同样的方式，可以从组件栏中选择静态文本及弹出式菜单对象来显示所需要处理的数据对象。此外，从组件栏中选择轴对象放在图形用户界面中。将以上几个组件都放置在图形用户界面后，可以调整这些对象的位置及大小。

调整完毕的布局图如图 17-31 所示。



图 17-30 “对齐对象”对话框

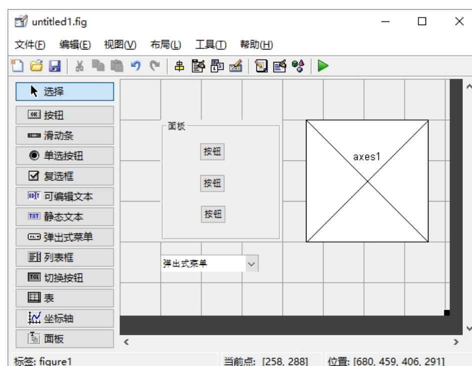


图 17-31 调整完毕的图形用户界面

可以从图形用户编辑界面中单击运行按钮来运行编辑过的图形用户界面（GUI）。此时，系统会提示将所创建的界面以.fig 的形式保存并打开保存对话框，输入名称 ex17_6 后保存。系统将会依次显示 M 文件编辑器及所编辑的图形用户界面，其中图形用户界面的运行效果如图 17-32 所示。

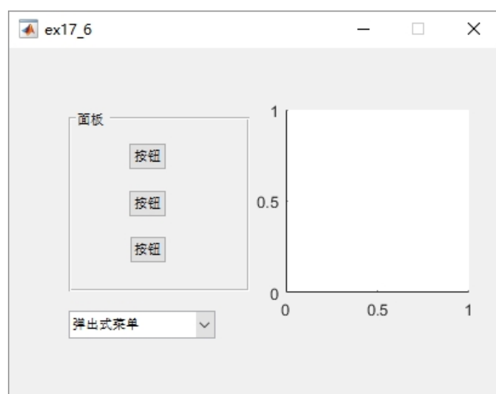


图 17-32 运行的图形用户界面

3．设置组件的属性

如果属性检查器已经关闭，则可以按照前面介绍的方法从工具栏中选择图标打开属性检查器。选择第一个按钮对象，在属性检查器中改变该按钮的属性。从属性检查器中找到第一个按钮的 String 属性，将其修改为 Surf，如图 17-33 所示。

以同样的方式，将第二个按钮和第三个按钮的 String 属性分别修改为 Mesh 及 Contour。选择面板后，将面板的 Title 属性修改为 Plot types。

编辑修改后的图形界面如图 17-34 所示。

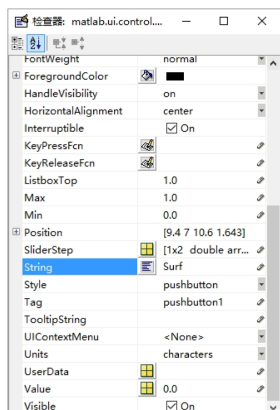


图 17-33　修改第一个组件的显示标签

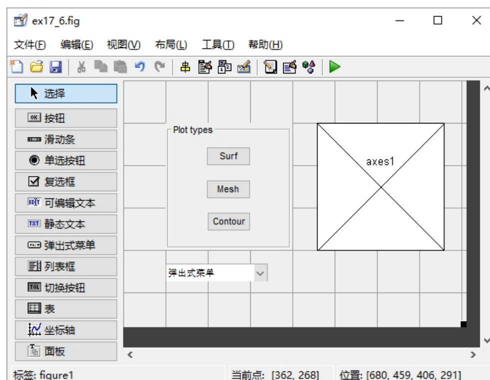


图 17-34　改变属性后的图形界面

下面对弹出式菜单进行设置。从属性检查器中选择 String 属性，输入可以处理的 3 个数据名称，即 Peaks、Membrane 和 Sinc，作为后面进行数据处理的对象。输入后如图 17-35 所示。单击“确定”按钮完成对弹出式菜单的设置。

此时，单击运行按钮▶，效果如图 17-36 所示。

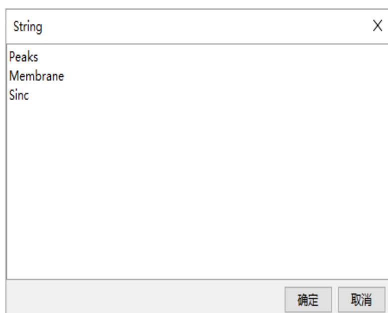


图 17-35　设置弹出式菜单项

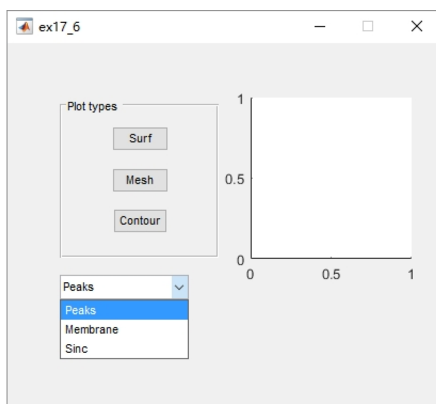


图 17-36　效果总图

4．编写回调函数

在完成上面的图形用户界面的编辑后，需要添加回调函数及一些相应的执行代码，所创建的图形用户界面才能完成图形函数的处理过程。因此，此时需要切换到 M 文件编辑器中进行函数文件的编辑。

当切换到该函数体后，可以输入以下语句来进行程序执行前的初始化。修改后的函数内容如下：

```
function ex17_6_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ex17_6 (see VARARGIN)
handles.peaks=peaks(35);
handles.membrane=membrane;
[x,y]=meshgrid(-8:.5:8);
```

```
    r=sqrt(x.^2+y.^2)+eps;
    z=sin(r)./r;
    handles.sinc=z;
    handles.current_data=handles.peaks;
    surf(handles.current_data);
    % Choose default command line output for ex17_6
    handles.output = hObject;

    % Update handles structure
    guidata(hObject, handles);
```

接下来的任务是为 3 个绘图按钮编写回调函数代码。这 3 个函数的回调函数代码可以直接在 M 文件编辑器的函数显示列表中选择；或者在 GUIDE 编辑器中，在对应的按钮上单击鼠标右键，从弹出的快捷菜单中选择“添加回调函数”命令切换到相应的 M 文件编辑器位置。

对于第一个按钮，在 M 文件编辑器的函数位置处添加以下代码：

```
    surf(handles.current_data);
```

以同样的方式为其他两个按钮添加回调函数代码：

```
    mesh(handles.current_data);
    contour(handles.current_data);
```

下面对弹出式菜单添加回调函数代码。以同样的方式，可以修改回调函数为：

```
    function popupmenu2_Callback(hObject, eventdata, handles)
    % hObject    handle to popupmenu2 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2 contents as
cell array
    %        contents{get(hObject,'Value')} returns selected item from popupmenu2
    Val=get(hObject,'Value')
    strl=get(hObject,'String')
    switch strl{Val};
    case 'Peaks'
      handles.current_data=handles.peaks;
    case 'Membrane'
      handles.current_data=handles.membrane;
    case 'Sinc'
      handles.current_data=handles.sinc;
    end
    guidata(hObject.handles)
```

上面的回调函数在选择弹出式菜单中的不同选项时，将会把不同的数据赋给当前的处理数据句柄。因此，在进行绘图时，将会按照要求进行处理。

5. 图形用户界面的执行

此时，运行处理程序得到的结果如图 17-37 所示。当然，用户可以选择其他的数据和绘图类型进行分析。

本节所修改的部分已经在前面分别列出。对于其他部分的代码由系统自动生成，因此，在一定程度上减小了程序编写的强度，使用户可以把更多的精力放在界面绘制和重要部分代码的编写上。

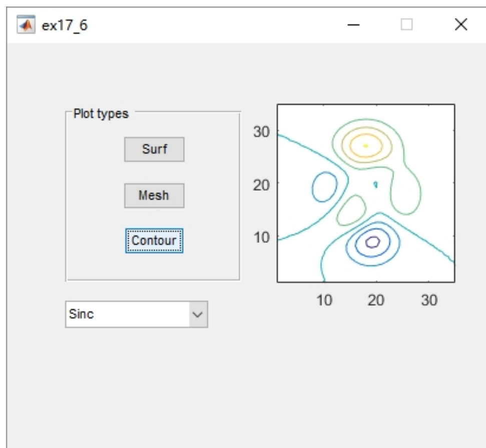


图 17-37　选择其中一个工具进行处理的结果显示（mesh）

17.6.2　GUIDE 创建的工具

在使用 GUIDE 创建图形用户界面时，还有一些比较常用的功能，能够更好地帮助用户完成图形用户界面的绘制过程。此处将对这些功能进行简单的介绍，读者在使用 GUIDE 创建图形用户界面时可以更方便地完成绘制过程。

1．位置调整工具

如果在创建过程中已经在 GUIDE 编辑界面中添加了多个组件，那么常常需要排列这些组件对象以完成界面的布局和排布。此时，用户可以从菜单项中选择工具→对齐对象命令，或者直接单击工具栏中的图标，系统将会弹出“对齐对象”对话框。

使用该对话框中的排列布局功能可以很快完成排列，如水平和竖直方向的对齐、分布间距等。选择排列对象时可以同时接下 Ctrl 键，用鼠标左键来选中每个需要进行排列的对象。如果需要进行微调，则可以在选择对象后，通过键盘方向键来实现。

2. 网格和标尺

选择此功能后，用户可以改变网格线的间距或显示标尺进行调整。选择工具→网格和标尺命令，弹出如图 17-38 所示的设置对话框。

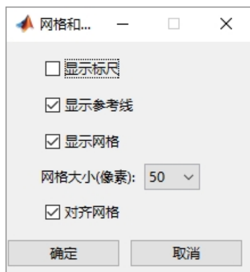


图 17-38　网格和标尺设置

在该对话框中可以通过选中复选框控制是否显示标尺、显示向导、显示网格间隔及是否吸附到网格。如果选择“显示网格”复选框，那么可以设置网格线之间的间隔，间隔的大小

可以从弹出式菜单中选择，间隔大小的单位为像素。

3．对象浏览器

对象浏览器可以帮助用户查看放置在图形界面中对象的名称及相互之间的继承关系等。可以直接单击工具栏中的“选择对象浏览器”图标来打开“对象浏览器”对话框，如图17-39 所示。

4．GUI 组件选项

从菜单栏中选择工具→GUI 选项命令，打开“GUI 选项”对话框，如图 17-40 所示。

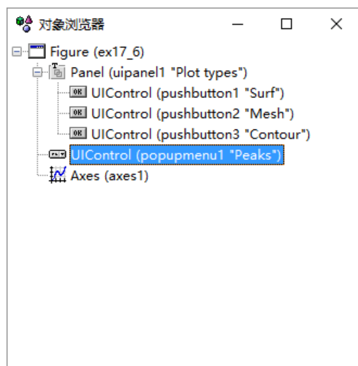


图 17-39　对象浏览器

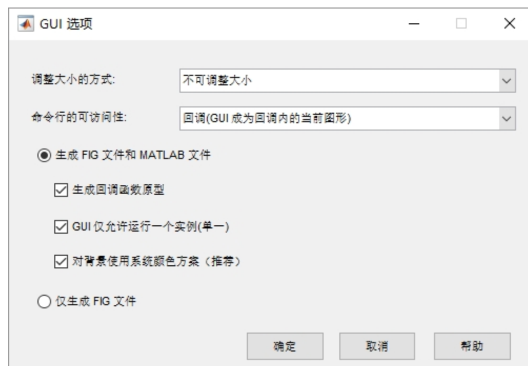


图 17-40　“GUI 选项”对话框

在该对话框中可以设置图形用户界面的一些属性。

（1）调整大小的方式。

- 不可调整大小：表示不可以调整图形界面的大小。
- 比例：表示按比例缩放图形对象的大小。
- 其他：表示根据 ResizeFcn 来调整大小。

（2）命令行的可访问性。如图 17-41 所示，在创建图形对象的过程中，选择是否通过命令行方式来访问各种对象。

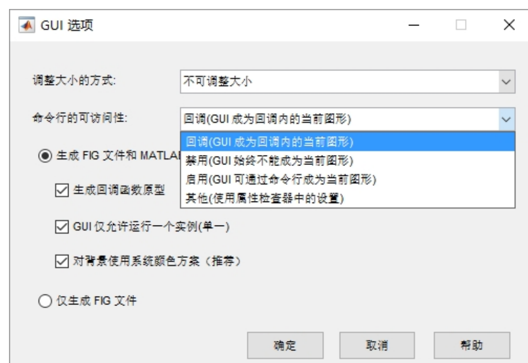


图 17-41　命令行访问的选择

- 禁用：禁止对 GUI 图形对象的访问，此时图形对象的句柄是隐藏的，通过对象句柄结构体来保存 GUI 中的所有用户控件对象句柄。
- 启用：可以在命令方式中对图形对象进行访问。

- 回调：可以通过回调函数来调用。
- 其他：可以使用属性检查器中的设定来访问。

在属性检查器中可以设置的两个属性为 Handle Visibility 及 lntergerHandle，前者决定图形窗口的句柄是否在命令行可见，如果设置为 Off，那么句柄将从根对象列表中删除，但句柄仍然有效；后者表示运行的图形窗口句柄是整数还是浮点数，如果设置为 Off，那么将会使用浮点数来代替整数。

（3）生成 FIG 文件和 MALTAB 文件。此项可以设置用户在创建 GUI 对象时，是否生成 FIG 文件和 M 文件。可以选择的设置选项主要有 3 个：

- 生成回调函数原型：此时将会自动在 M 文件中为每个组件添加一个回调函数原型。
- GUI 仅允许运行一个实例。此选项表明每次运行时，只允许一个示例运行。如果对象示例已经存在，那么将会带到前台。如果取消选择该复选框，那么每个调用命令都将会产生一个 GUI 图形对象示例。因此，多次运行将会产生多个运行实例。
- 对背景使用系统颜色方案。此时将会使用系统的颜色方案保持颜色一致。

（4）仅生成 FIG 文件。此时用户仅仅创建 FIG 文件，可以使用 open 或 hgload 来显示该文件。一般在用户希望创建一个与 M 文件完全不同的实例时，可以选择此选项。

17.6.3　创建带 Uicontrol 控件的图形界面

此处介绍以使用 GUIDE 的另一个模板方式创建的用户图形界面，即带 Uicontrol 控件的图形用户界面。以同样的方式打开 GUIDE 界面，选择 GUI with Uicontrols 方式来创建图形界面，如图 17-42 所示，此时在右侧出现模板图形预览。

在该对话框中单击“确定”按钮，打开如图 17-43 所示的图形用户界面，在该图形界面中可以根据输入的不同单位值数据计算质量。

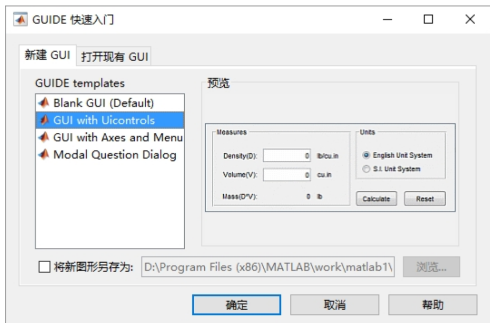


图 17-42　创建带 Uicontrol 的图形界面

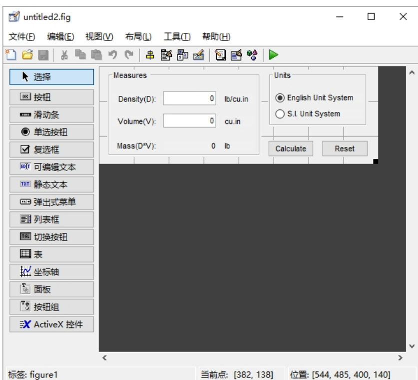


图 17-43　带 Uicontrol 的图形界面

单击对象浏览器，可以查看该图形界面的所有对象及其相互之间的继承关系。从图 17-43 中可以看出该图形界面由两个面板组成，分别为测量（Measures）面板和单位（Units）面板。在这两个面板中可以输入需要计算的数据及单位制的选择。此外还包括两个按钮：计算

（Calculate）和重置（Reset）。

图形对象浏览器如图 17-44 所示，将上面的图形界面及 M 函数文件同时保存，文件名为 GUIDE_uicontrol_exam。此时，可以切换到 M 函数文件浏览器来查看所有输入的函数文件，单击“显示函数”弹出式菜单即可看到所有的函数。

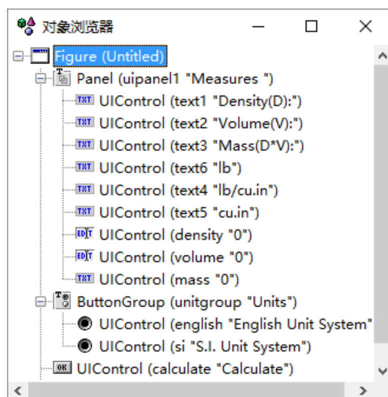


图 17-44　图形对象浏览器

接下来的任务是在图形界面编写回调函数代码和相关程序语段及保存运行。回调函数的代码是基于具有 MATLAB 编程基础的读者根据图形界面所需功能效果的实现而编写的相应内容。

本节简单介绍了使用 GUIDE 的 Uicontrol 模板来创建用户图形界面的一些方法。实际上使用该模板来创建图形界面的方法和使用空白模板的创建方法相同，只是采用该模板后提供了一些控件，可以更方便地完成处理过程。

17.7　本章小结

MATLAB R2018a 提供了大量的图形用户界面处理函数，能够帮助用户创建友好的图形用户界面，还能够帮助用户更好地和 MATLAB R2018a 的处理程序交互。在本章中，对使用函数方式创建图形用户界面的方法进行了介绍，包括图形用户界面控件的创建、鼠标动作执行、事件队列的执行顺序、回调函数的编写及对话框对象、界面菜单等。这些内容是使用函数来创建图形用户界面的基础。